vmware PRESS

VMware NSX® Network Virtualization Fundamentals

Gustavo A. A. Santana, VCIX-NV® and CCIE® 8806

Foreword by Pat Gelsinger

VMware NSX® Network Virtualization Fundamentals

Gustavo A. A. Santana, VCIX-NV® and CCIE® 8806

Foreword by Pat Gelsinger

vmware® PRESS

VMWARE PRESS

Program Managers

Katie Holms
Shinie Shaw

Technical Writer

Rob Greanias

Graphics Manager

Sappington

Production Manager

Sappington

Warning & Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The authors, VMware Press, VMware, and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

The opinions expressed in this book belong to the author and are not necessarily those of VMware.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA
Tel 877-486-9273 Fax 650-427-5001 www.vmware.com.

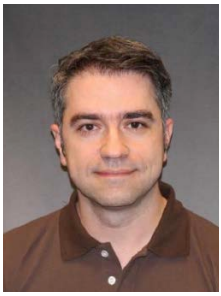
Table of Contents

List of Figures

List of Tables

List of Examples

About the Author



Gustavo A. A. Santana is the leader of the Software Defined Data Center (SDDC) systems engineering team in VMware Latin America and the author of two books: Data Center Virtualization Fundamentals (Cisco Press, 2013), and CCNA Cloud CLDFND 210-451 Official Cert Guide (Cisco Press, 2016).

With more than 20 years of experience in the data center industry, Gustavo has worked on multiple enterprise and service provider data center projects that required extensive integration among multiple technology areas such as networking, application optimization, storage, and servers.

A true believer of education as a technology catalyst, he has also dedicated himself to the technical development of IT professionals from multiple customers, partners, and vendors. In addition to holding the VMware Certified Implementation Expert in Network Virtualization (VCIX-NV) certification, Gustavo is a triple Cisco Certified Internetwork Expert (CCIE), and an SNIA Certified Storage Networking Expert (SCSN-E).

A frequent speaker at VMware and data center industry events, he holds a degree in computer engineering from Instituto Tecnológico de Aeronáutica (ITA-Brazil) and an MBA in strategic IT management from Fundação Getúlio Vargas (FGV-Brazil).

About the Technical Reviewer

Gabriel Maciel is a former police officer with more than 18 years of experience in information systems, networking & security, and IT systems architecture. Gabriel leads the SLED Networking & Security Systems Engineering specialist team at VMware. He is a regular speaker at national and international events where he shares his technical experience with clients, end users, and enterprises. His background includes technical and managerial positions in higher education, public sector (e.g., state/local, federal, and defense), and financial services.

Dedications

First and foremost, this book is dedicated to my wife and true love, Carlene, whose patience and humor has helped her define my writing process as “productive aloofness”. Besides supporting professional endeavors such as this, she is my co-architect and joint operational manager on two joyful projects called Carolina and Cecília.

I also dedicate this publication to my parents, Honorio and Cleia, who have taught me, my brother André, and my sister Raquel, that one can only learn by being fearless and humble.

And finally, this book is dedicated to all IT professionals that were curious enough to question their common believes to embark with us on an exciting journey called network virtualization.

Acknowledgements

This book is the direct result of the two most exciting years of my engineering career. Only through uncountable discussions, laboratories, proofs-of-concept, and design workshops was I able to create and structure this writing. So here I acknowledge the amazing network of professionals that have positively contributed to its content.

First, I heartily thank the Latin America NSX Systems Engineering team: Andres González, Caio Oliveira, Daniel Aguirre, David Santos, Gonzalo Atienza, Irapuan Lima, Jorge Hernandez, José Zavala, Waldemar Pera, and Thiago Koga. Their technical expertise, customer focus, and collaborative work have driven the wave of change that is still rippling strongly in the region.

I would also like to express my gratitude to my friends and trusted advisors: André Andriolli, who convinced me that an open mind is the best asset to any career; and Alexandre Moraes, whose advice as an author has always kept me off the beaten path.

My sincere thanks to Paul Byrne, for making me realize that being a leader is serving, and also for helping me coordinate my professional life and this writing.

A very special thanks to Gabriel Maciel, my friend and brilliant technical reviewer.

I am also very grateful for Dom Delfino, Steve Tegeler, and Rob Randell for making me feel at home since the very first interview in VMware.

Many thanks to all VMware Press production team, especially Katie Holms, Shinie Shaw, Kausum Kumar, and Rob Greanias.

And finally, I dearly appreciate the effort, dedication and openness of more than one hundred VMware LATAM partner engineers that actively participate on the Delta Force NSX program.

Foreword

As with most infrastructure technologies of the ‘80s and ‘90s, networking was intrinsically architected to single-purpose proprietary hardware since its inception. In the last 20 years, new application architectures have changed the landscape from data centers to centers of data utilizing the unlimited scale of the Cloud. The intelligent and secure transport of data required a radical new approach.

Back in 2010, I was introduced to a particularly innovative distributed software architecture that decoupled the network intelligence from the underlying network transport. This was in a meeting with Martin Casado and the team from Nicira. After my 30-year career at Intel, it became very clear that Nicira’s distributed systems architecture was able to leverage the ever-increasing power and ubiquity of the x86 platform. This was the right bet at the right time to transform networking.
At last, networking functions could be fully uncoupled from physical dependencies.

Fast-forwarding 8 years, VMware NSX – the result of the acquisition of Nicira - can be undoubtedly considered the first and most fundamental step into VMware’s vision for the future of networking: The Virtual Cloud Network. Currently providing application security, traffic visibility, and network agility to thousands of organizations around the world, NSX materializes the Virtual Cloud Network as a pervasive and secure software layer that offers end-to-end connectivity for applications and data, wherever they are – data center, cloud, or edge.

In this book, Gustavo Santana - a true network expert and enthusiast I met on my first trip to Brazil – introduces the most fundamental concepts of network virtualization in a personable and lively way. With more than 20 years of technical experience in networking, Gustavo explores the many challenges network and security professionals face every day and how NSX applies such concepts to overcome these obstacles with elegance and scale.

So, welcome to the network revolution and make yourself at home. As you start reading, you will be able to envision the huge opportunity many IT professionals are already leveraging in their own digital career transformation.

Pat Gelsinger
Chief Executive Officer of VMware

Abstract

During their digital transformation process, many IT organizations still struggle with traditional networking methods and security approaches. By successfully addressing these challenges in thousands of real-world implementations, VMware NSX has established itself as the leading network virtualization platform, revolutionizing the way data center networks are designed and operated.

In this book, data center expert and author Gustavo A. A. Santana thoroughly examines the specific circumstances that created such challenges and explains how VMware NSX overcomes them. This context will help those who want to:

- Understand the difference between VMware NSX network virtualization and other virtualization techniques in networking
- Comprehend how relatable network constructs such as logical switches, distributed logical routers, and edge service gateways can radically simplify network provisioning
- Substantially improve data center security with the Zero Trust model while implementing micro-segmentation, scalable virtual private networks, and integration third-party solutions
- Easily extend virtual networks between two or more data center sites to deploy active-standby or active-active business continuance scenarios
- Smooth data center automation and make private cloud computing implementations possible
- Radically simplify data center physical network architectures and enable full traffic visibility over physical and virtual network devices

Right from the start, Santana explores key networking concepts and how they are seen through the virtualization lens. This book is an essential resource for any IT professional that wants to understand the full potential of an enterprise-class network virtualization platform such as VMware NSX.

Introduction

If content is king, context is the kingdom.

While many excellent books about VMware NSX already exist, I fully intended to write this one with an unorthodox approach. Having participated in literally hundreds of discussions around network virtualization in the last two years, I have purposely planned it to be primarily focused on the whys related to the theme. In other words: the practical reasons that justify why so many organizations are using this solution.

In order to address a satisfying amount of hows and whats to an intended audience, I usually design publication content with certain individuals in mind. However, I wrote this book thinking about someone new: me. More specifically, myself before meeting Martin Casado in 2014 during his first trip to Brazil.

Back then, the father of software-defined networking (SDN) was happily discussing with me what this brilliant new product could potentially achieve and how it was effectively helping early-adopting organizations in many other countries. During the conversation, as a networking professional with more than 15 years of experience, all I could ask were implementation details such as encapsulation protocols and multi-destination traffic methods.

But thanks to Martin's patience – and also some introspection time – I was able to realize that my own specialization was making me shortsighted to the inadequacies of the networking status quo at that period. In fact, I was learning that complexity, which was a vast part of my job, was exactly what data center architects were trying to get rid of.

To say that this contemplation changed my career is a gross understatement. Because it has molded me to approach projects in a new way, I was able to write each book chapter to zoom in on challenges that modern data centers face when dealing with long-established networking and security approaches. And with such backdrop, the book intends to clarify the ways NSX leverages network virtualization to address these conundrums.

For the sake of illustration, here is a brief description of the book chapters:

- **Chapter 1, "What Is Network Virtualization?":** If virtualization has commonly taken part of many techniques in networking, what is so special about NSX? After answering this question, this chapter examines what is expected from this platform and also details of its internal architecture.

- **Chapter 2, "Rethinking Network Constructs":** Seasoned data center professionals are used to configuring networking services such as VLANs, subnet addressing, dynamic routing, and server load balancing. NSX reimagines these components through simpler-to-provision virtual constructs such as logical switches, distributed logical routers, and edge gateway services.

- **Chapter 3, "Virtualization as a Security Control":** Data center security has always been closely associated with networking concepts such as firewall rules, demilitarized zones, VLANs, and access-control lists. Nevertheless, modern cyber-attacks have already assimilated all vulnerabilities associated with this traditional model, requiring new security approaches such as micro-segmentation and the Zero Trust model. NSX brings these propositions to reality with features such as distributed firewalling and Service Composer.

- **Chapter 4, "Virtual Networks, Real Extensions":** The extension of networks between distinct data center sites has generated more friction between operational teams than some democratic elections. Network virtualization presents an alternative that addresses this challenge with simplicity and familiarity to these contending areas.

- **Chapter 5, "Industrializing Networks":** Automation is definitely a buzzword that is commonly discussed but seldom implemented. In fact, most IT professionals who have dealt with network automation have more bad stories to share than accomplishments. Network virtualization introduces the definitive model of on-demand networking and security services for automated environments.

- **Chapter 6, "One Network":** The most successful digitally-transformed organizations have abandoned the physical versus virtual discussion to focus on the design and operation of their data center network as a single entity. This chapter discusses how this approach is already preparing data center architects for the next evolutionary step of application development.

At the end of this almost 200-page journey, I truly hope to have offered a new perspective of how exciting and powerful networking still is, especially when virtualization concepts are fully applied to it. If you are interested, I'll meet you and your open mind in the next page.

What Is Network Virtualization?

> *"We all live every day in virtual environments, defined by our ideas."*
> *Michael Crichton*

In times when even human interactions can be virtual, does it still make any sense to celebrate a new virtualization technology as revolutionary?

As thousands of organizations can attest, VMware NSX has firmly established itself as the leading network virtualization platform in the world since its first release in 2013. With more accelerated market traction than any other virtualization solution - even above the company's previous forerunner, VMware vSphere® - NSX has proven how networking and security are still critical to the business outcomes of digitized corporations while also making use of the wide-open space for improvement in modern data centers architectures.

Nevertheless, a cultural shift is always slower than a technological one. In a world of big data and fast information, many networking and security professionals still struggle to grasp the true value of *network virtualization* among the many fleeting innovations-of-the-week. In fact, this confusion is fairly understandable because the term *virtualization* has been consistently used to describe many networking technologies since the humble beginnings of this area of information technology expertise.

Unceremoniously leveraging many candid discussions held in the last two years with customers, partners, and VMware colleagues, this chapter provides an analytical comparison between some of the most common virtualization technologies and the *network virtualization* enabled by NSX. Drawing a parallel between the server virtualization revolution in the mid-2000s and NSX, it uncovers how this virtualization platform fits into a data center network infrastructure as well as some relevant details of its internal architecture.

Virtualization as the New Reality

Within the information technology market, VMware is considered the undisputed worldwide leader of virtualization technologies. It provides virtualization solutions to a staggering number of computer systems, including high-end servers, personal computers, and mobile devices.

Although VMware is a relatively young company - entering its twenties in the same year of this writing - the term *virtualization* has been persistently used in computing lingo for more than 50 years. As a support for this statement, the following list compiles some computing technologies that have relied on this word to describe their core concepts:

Time sharing: Created in 1957 at Stanford University, this technology divides the processing resources of a computer among multiple programs, or mainframe users back when the technology was invented. It offers an equal slice of time to each program through the halting of program execution, saving its state in memory while loading and running another program state. With such arrangement, each program has the illusion of control over the computer operating system.

Virtual memory: Conceived in 1959 at the University of Manchester, this virtualization technique uses auxiliary memory peripherals as an extension of the main memory. This is faster, but also shorter and more expensive than the former. Using virtual memory, a computer central processing unit (CPU) can directly access a virtual memory address unbeknownst of its real location.

Virtual machine: Released by IBM in 1972, this technology was based on the emulation of the mainframe architecture, allowing an operating system to run transparently over a software-emulated mainframe hardware. In this architecture, a *control program* (CP) controlled the mainframe hardware, dedicating a share of hardware resources to conjure an abstraction where each mainframe user had the perception of accessing an exclusive operating system for their own purposes.

Redundant Array of Independent Disks (RAID): Developed in 1987 at the University of California, Berkeley, this popular virtualization technology aggregates data blocks from multiple hard disk drives to achieve storage high availability, performance, or simply greater capacity. Moreover, a *RAID group* produces the illusion of a single virtual hard disk drive.

Even though this list can be certainly much longer, these brief descriptions already present some of the characteristics that all virtualization technologies share.

It is noticeable that these techniques are essentially an *emulation* of a computing resource - computer operating system, primary memory, mainframe hardware, and a hard disk drive, respectively - in order to give the *perception of a real resource* to a *consumer* - program, computer CPU, mainframe operating system, and a computer.

Furthermore, all of these abstractions are implemented through *software* and bring *advantages* that were not possible for their emulated counterparts - multitasking, primary memory extension, multiple operating system versions, storage availability and scaling.

Thereupon, this writing will use the following formal definition of virtualization to avoid confusion with any preconceptions:

"Virtualization is the faithful reproduction in software of an IT resource delivering to its consumers benefits that were unavailable in its physical form."

Generically speaking, the biggest accelerator of any virtualization technique has been its capability of leveraging processes and technical skills that were previously defined for the emulated physical resources. Undoubtedly, such similarity drastically reduces cost and complexity in implementation.

But if virtualization has such a long history in computing, why are CIOs still pursuing it so vigorously in their IT environments? And why is virtualization still such a strong buzzword after half a century of permanence in computing?

The answers to these - and many other philosophical questions - will be supplied in the next section.

The Server Virtualization Revolution

A *server* can be defined as a computer system which holds an application that receives requests from multiple clients, processes them, accesses other servers, and subsequently provides appropriate responses to these clients.

In the early 2000s, most data centers had already transitioned their compute architecture from mainframe computers to microcomputers based on Intel x86 CPUs. They had securely migrated the large majority of their mission-critical applications to this open, flexible, and affordable processing platform. Consequently, with x86 servers increasing their relevance to business, computer vendors started to commercialize specialized hardware that was better suited to run these central application components.

At that time, if a new application was required, it was common to deploy *a new dedicated server computer* to run this software. Such an approach was largely adopted to avoid application performance issues. Hardware resources were sized to support processing peaks that could happen daily, weekly, monthly, or even yearly for some applications.

As more applications were deployed in data centers, operations teams supporting servers realized that these devices were poorly used during off-peak periods. This affected the capacity planning of other technology areas; a server working at 10 percent of its capacity still consumes 100 percent of its space, storage, cabling, and network ports.

At this point, VMware had already successfully brought the concept of the mainframe virtual machine to x86-based computers. While it generated great interest to desktop users, the company achieved a significant breakthrough in 2003, after it released a product called *VMware ESX®*.

In a nutshell, this software allowed server applications to run on virtual machines, achieving a performance comparable to the same applications running on physical servers.

Figure 1-1 shows a process called *hardware consolidation* that helped many data centers avoiding resource depletion due to the one-application-per-server model.

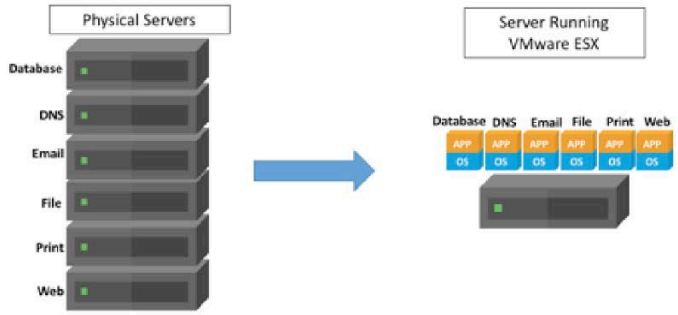


Figure 1.1 Hardware Consolidation through Server Virtualization

With the migration of applications from physical servers to virtual machines, data centers achieved a more efficient use of x86 servers as well as all other solutions that supported them - such as facilities, storage, and networking.

VMware ESX is a *hypervisor*, which is formally defined as an application that can create virtual hardware including:

- Virtual CPUs
- Virtual memory
- Virtual hard disk drives
- Virtual network interface controllers
- Virtual peripherals (e.g., CD-ROMs)

These virtual emulations are jointly used for the installation of a *guest operating system* and server applications. Therefore, in the context of modern server virtualization solutions, a *virtual machine* is defined as the set comprising virtual hardware, guest operating system, and associated applications.

Note: Not all hypervisors are alike. According to a classification system developed by Gerald J. Popek and Robert P. Goldberg in their 1974 article "Formal Requirements for Virtualizable Third Generation Architectures", there are two types of hypervisors. A Type-1 hypervisor replaces the operating system as the software component that directly controls the hardware. For this reason, it is also known as native or bare-metal hypervisor. On the other hand, a Type-2 hypervisor runs over a preexistent operating system. This is also referred to as a hosted hypervisor, and is expected to perform below the level of Type-1 hypervisors. VMware ESX was the first Type-1 hypervisor for x86-based computers.

VMware ESX became a huge success, enabling virtual machines to replace physical servers as the new *atomic units* of data centers. With the intention to ease the control of these ever-sprawling virtual servers, VMware released the concept of a *server virtualization platform*. This included a new, leaner version of its hypervisor (i.e., ESXi) and a comprehensive set of management tools such as *Virtual Center Server* - later renamed VMware *vCenter Server®*.

Distinctive from a standalone virtualization technology which has a predefined benefit over its physical counterpart, a *virtualization platform* allows the development of features and advantages that were not originally envisioned in its initial conception.

Initially called *VMware Infrastructure*, VMware's server virtualization platform further evolved in 2009 with the launch of *VMware vSphere*. VMware vSphere continues to maintain its position as the leading server virtualization architecture with approximately 80% of the hypervisor market[1].

Figure 1-2 examines the main components of VMware vSphere.

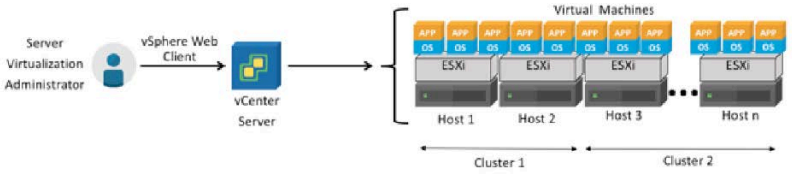


Figure 1-2 VMware vSphere Architecture

As an example of a *Virtual Machine Manager*, vCenter Server is a software solution that can create and manage virtual machines running on physical servers with VMware ESXi hypervisors. These devices, which are commonly called *hosts*, can be grouped into *virtualization clusters* to share the same policies and feature settings defined by vCenter. In order to carry out these operations, server virtualization administrators use the *VMware vSphere® Web Client* to access vCenter.

VMware vSphere brought more benefits to data centers with each of its new versions. As a true server virtualization platform, vSphere enabled a large number of features - developed by VMware and third-party technological partners - that were beyond the wildest dreams of any physical server administrator.

Table 1-1 describes some of the most interesting VMware vSphere features.

[1] http://www.vmware.com/company/why-choose-vmware/sddc-platform.html

Table 1-1 VMware vSphere Virtualization Features

Feature	Description
High Availability (HA)	After a host experiences a major failure, its virtual machines can be automatically restarted on other hosts from the same cluster.
vMotion	Enables the migration of a virtual machine between two hosts with minimal disruption in its guest OS and hosted applications. vCenter coordinates a copy of a virtual machine to another host until all of its characteristics are synchronized. Then, the original VM is erased and the copy is fully activated.
Distributed Resource Scheduler (DRS)	According to a preconfigured policy, hosts that may be on the verge of a utilization threshold can *preemptively migrate* virtual machines to other hosts. This helps optimize utilization across a virtualization cluster via distribution of VMs.
Fault Tolerance (FT)	Provides continuous availability for virtual machines by creating and maintaining a secondary VM. This identical VM is continuously available to replace the primary VM in the event of a failover situation.
Distributed Power Management (DPM)	VMware vCenter calculates the amount of resources used by all active VMs and analyzes whether some hosts on the virtualization cluster can be put in "standby mode". If so, those VMs are migrated to a subset of the cluster, enabling automatic power savings during off-peak periods.
Maintenance Mode	If a host from a virtualization cluster requires a disruptive operation (e.g., ESXi version upgrade), an administrator can activate this mode to automatically migrate VMs to other hosts in the cluster and prohibit any VM-related operation on this host.
Snapshot	Preserves the state and data of a VM at a specific instant so the virtualization administrator can restore that VM to its exact snapshot state.
Cloning	Creates a copy of a VM resulting in a completely independent entity. To avoid future conflicts, the clone VM is assigned different identifiers such as name and hardware addresses.
Template	A non-productive VM that will be cloned frequently in order to accelerate the deployment of applications. A template provides a more secure replication solution than a clone because it cannot be changed as easily as a standard VM.

Features were continuously incorporated into VMWare vSphere, simplifying and streamlining application deployment and server monitoring in most data centers. Invariably, this server virtualization platform transformed these environments into centrally-managed dynamic structures, paving the way to innovative concepts such as IT automation and cloud computing.

Virtualization in Network Technologies

The use of the word *virtualization* is not restricted to computing technologies. Seasoned IT professionals are aware that the transparent emulation of network constructs is a traditional technique that is present on most network devices.

Further developing this statement, the next sections will review virtualization technologies that have been commonly deployed in data center networks for some decades.

Virtual Local Area Networks

A *local-area network* (LAN) is a computer network that interconnects hosts in a limited area such as a building or campus. Because LAN connections are relatively short, these networks use distinct data transmission technologies from *wide-area networks* (WANs).

Although many other technologies have been used since its inception in the 1970s, LANs primarily use the Ethernet protocol to exchange data between computers and network devices. The most common devices on a LAN are *switches*, which forward Ethernet frames based on their media access control (MAC) addresses via hardware-based processes embedded into application-specific integrated circuits (ASICs).

Note: To learn which MAC addresses are connected to its interfaces, an Ethernet switch leverages a process called *transparent MAC learning.* Entries are inserted in the MAC address table every time an unknown MAC address is detected as the source of an Ethernet frame or a known MAC address is detected on a different interface. Each entry is maintained during a defined *aging time* interval and is automatically deleted if other frames with such address are not detected.

Broadcast communication is commonly expected in Ethernet LANs transporting Internet Protocol (IP) packets. Whenever a switch interface VLAN receives a broadcast Ethernet frame (i.e., destination MAC address ffff.ffff.ffff), the device must forward this frame to all other ports. However, if an excessive number of hosts are connected to the same LAN, it may introduce communication problems such as a *broadcast storm*.

With the objective of preventing every LAN from being a single shared broadcast domain, computer scientist Walter David Sincoskie created the concept of a *virtual local-area network* (VLAN) while working at Bellcore in 1984. A VLAN can be defined as an administrator-configured broadcast domain than exists in a single Ethernet switch or is shared among connected switches.

Figure 1-3 illustrates this behavior in an Ethernet switch provisioned with three VLANs.

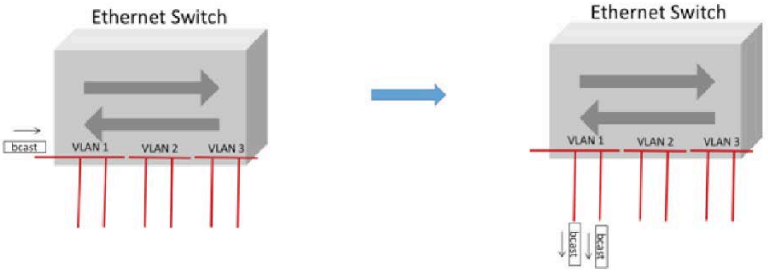


Figure 1-3 VLAN Behavior with Broadcast Frames

As Figure 1-3 exhibits, each VLAN emulates independent Ethernet switches not sharing broadcast frames. Additionally, these virtual elements do not share any other type of traffic, providing an efficient method of network segmentation. As a realistic emulation of a switch, VLANs also forward Ethernet frames based on their destination MAC address.

Traditionally, VLANs are assigned to a set of interfaces on a switch. If an interface belongs to a single VLAN, it is commonly defined as an *access port.* Ethernet interfaces are called *trunks* if they are configured to transport multiple VLANs concurrently.

Trunks were invented to reduce the number of connections between network devices deploying multiple VLANs. In order to identify the VLAN that each frame belongs to when traversing a trunk, the sending network device tags each frame with its *VLAN identifier* (VLAN ID), which is read by the receiving network device.

Standard 802.1Q from the Institute of Electrical and Electronic Engineers (IEEE) formally regulates this tag format. It defines the VLAN ID field as 12 bits, resulting a theoretical limit of 4096 VLANs per local area network.

Note: By definition, VLAN IDs 0 and 4095 are not used. As many switches also reserve some VLANs for internal process communications and/or due to specific ASIC limitations, the number of available VLANs on a device can be smaller than this value.

Figure 1-4 depicts an example of a network topology deploying both access and trunk interfaces.

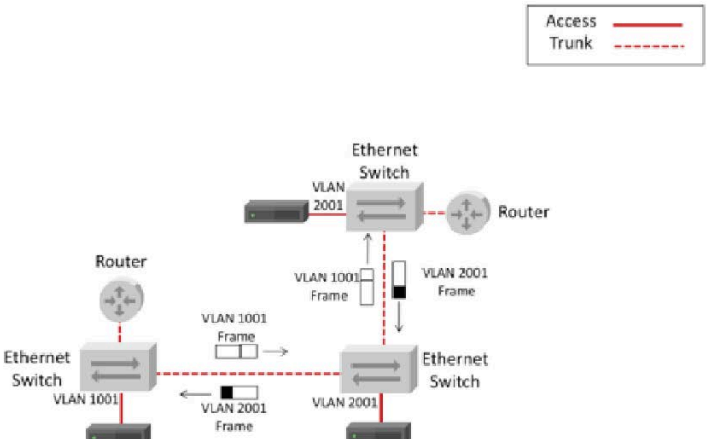


Figure 1-4 VLAN Topology

In Figure 1-4, access ports in VLANs 1001 and 2001 are used to connect servers to each of the shown switches. Trunks are used to connect switches and other IEEE 802.1Q-compatible devices such as routers. In this scenario, these layer 3 devices can deploy multiple *sub-interfaces* (one for each VLAN) on a single physical interface, comprising a routing design that was once called *router-on-a-stick*.

At first, routers connected to LANs in such fashion enabled the communication between devices connected to different VLANs. With time, network vendors introduced the concept of *layer 3 switches* – based on the Open Systems Interconnection [OSI] model routing layer - which could natively provide routing services between provisioned VLANs with more performance and availability.

Virtual Router Redundancy Protocol

In local area networks, a host relies on a router interface to forward packets to another IP subnet. The router interface IP address is registered on each host as its *default gateway*.

Because the great majority of operating systems only allow one default gateway definition per host, a potential connectivity breakdown may happen if the router that contains the defined gateway IP address - or its connection to the LAN - fails.

To address such single point of failure, a virtualization technique called *Virtual Router Redundancy Protocol* (VRRP) was standardized in the Internet Engineering Task Force (IETF) Request for Comments (RFC) number 2338 in 1998.

Figure 1-5 portrays the underlying foundation of this protocol.

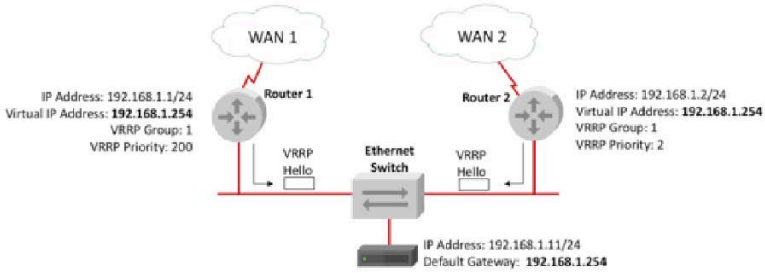


Figure 1-5 VRRP in Action

As the figure explains, VRRP provides default gateway redundancy through a *virtual IP address* that is shared between two or more router interfaces. Through the exchange of VRRP Hello messages, these network devices discover each other and decide which one assumes the virtual IP address for the local hosts using it as their default gateway.

This election process is defined via a preconfigured interface priority - higher value wins - or through a comparison of the IP addresses on each router interface – higher value winning again. After this process, one router has an *active* interface on this VRRP group, while all other interfaces deploying VRRP in this group remain in a *backup state*.

If an active interface fails, it will not send any VRRP hello messages to the other routers. A standby interface will become active and deploy the same virtual IP.

Virtual Routing and Forwarding

This virtualization technique creates multiple routing instances - called VRFs - that can coexist in the same routing equipment. Each VRF includes a separate routing table, a set of interfaces, and independent dynamic routing protocols such as Open Shortest Path First (OSPF) and Border Gateway Protocol (BGP).

Figure 1-6 depicts the creation of several VRFs on the same physical router.

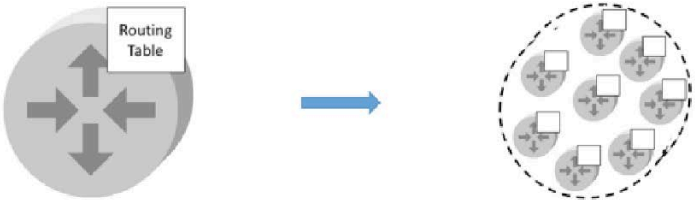


Figure 1-6 VRFs in Action

VRFs were originally introduced as a feature to Multiprotocol Label Switching (MPLS) in 1999, and are popularly used in pure IP-based networks for reasons such as:

- Network segmentation
- IP address overlapping
- Virtual Private Networks (VPNs)

Note: The section "The Data Center Network as a Service Provider Backbone" will provide more details about MPLS and VPNs.

Virtual Switching

Along with the introduction of virtual machines in x86-based micro-computers, VMware also fomented a quiet revolution through the concept of the *virtual switch*, a software abstraction where a hypervisor emulates an Ethernet switch in software to control VM traffic. The original networking element created by VMware was known as *vSphere Standard Switch* (vSS).

Figure 1-7 demonstrates how the vSS deals with traffic originating from and destined to virtual machines.

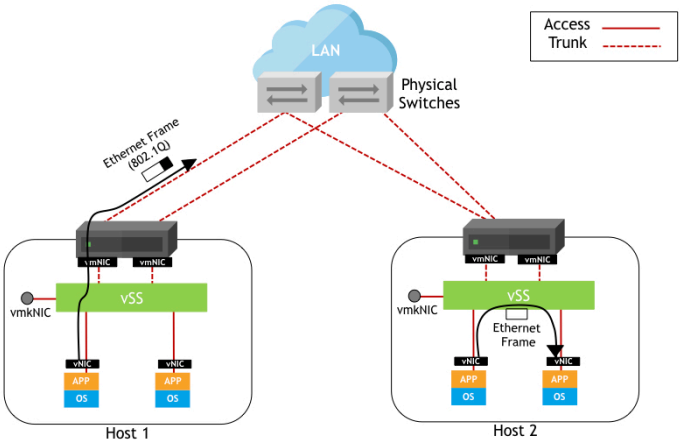


Figure 1-7 vSS in Action

A host's physical NICs act as the vSS *uplinks*, providing communication between the virtual device and the connected physical switches. As "The Server Virtualization Revolution" section discussed, each VM can have one or more virtual network interface controllers to provide bidirectional Ethernet connectivity.

In Figure 1-7, a vSS deployed on Host 1 can use this connection to forward Ethernet frames from a VM to the physical switch, and vice-versa. For Host 2, the data exchanged between the two VMs in the same host only requires a memory-based operation.

Figure 1-7 additionally includes some VMware vSphere networking definitions that are further explained in Table 1-2.

Table 1-2 VMware vSphere Virtualization Features

VMware vSphere Interface	Description
vmNIC	*Virtual machine network interface controller.* It represents the physical NICs on a host and performs the role of an uplink for the vSS.
vNIC	*Virtual network interface controller.* It embodies the emulation of a network adapter within a virtual machine.
vmkNIC	*Virtual machine kernel network interface controller.* A virtual interface with an IP address created in the ESXi kernel used for management, IP storage access, VM memory exchange during a live migration, and other functions.

Using the IEEE 802.1Q standard, the vSS identifies which VLAN an incoming Ethernet frame belongs to and signal to the access switch the VLAN for an outgoing frame. To assign a VM vNIC to a VLAN, vSphere uses a connectivity policy called *port group*, which also defines:

- **Security policies:** Rejects promiscuous mode - which can receive and process any Ethernet frame; accepts MAC address change detection and forged transmits - which are different from what is defined for the VM
- **Traffic shaping:** Defines average bandwidth, peak bandwidth, and burst size for outbound traffic
- **Physical NIC teaming:** Specifies load balancing algorithm, network failover detection method, switch notifications, failback, and network adapter fail order

In the case of the vSS, a port group is configured on a per-host basis. With time, the repetitive creation of these connectivity policies became a significant administrative burden in scalable server virtualization environments. Moreover, VMware® vMotion™ requires that both source and destination hosts have vSS and port groups named the same, increasing the probability of human error in these configuration procedures.

To bring relief to these vSS operational strains, VMware released a new virtual networking device in 2009, formally known as the *VMware vSphere® Distributed Switch™ (vDS).* This virtual switch is illustrated in Figure 1-8.

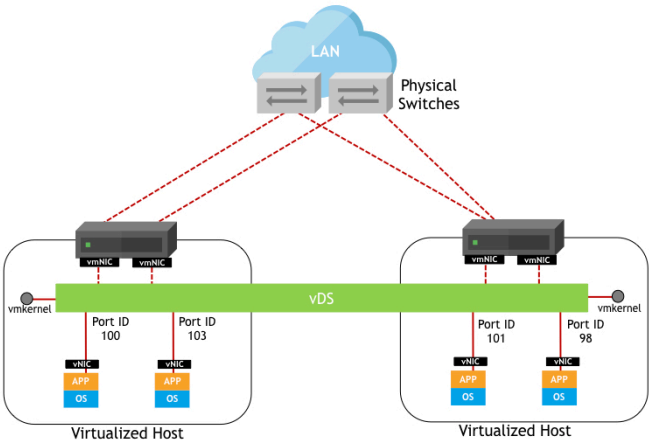


Figure 1-8 vDS in Action

In Figure 1-8, two hosts are connected to the same virtual networking device: a vDS. From an administrative standpoint, this perception is heightened via the creation of *distributed port groups*, which are configured once per vDS in vCenter and distributed to every host connected to such virtual device.

In addition to configuration optimization, vDS also introduced a series of enhancements over vSS, such as:

- **Uplink connectivity:** Deploy an explicit failover order to the connected vmnics; use hashing - over internal port ID, MAC address, or IP address - to load balance traffic to active uplinks; employ Link Aggregation Control Protocol (LACP) to negotiate aggregation with physical switches; or control upstream frame distribution via traffic load measurement.
- **Network Input Output Control (NIOC):** Protects and prioritizes traffic when there is capacity contention on the vDS uplinks.
- **Traffic Shaping:** Defines average bandwidth (Kbps), peak bandwidth (Kbps), and burst size (KB) per vNIC to provide different levels of service to virtual machines. Distinct from vSS, it can be applied ingress, egress, or in both directions on each vDS port.
- **Private VLANs:** A traditional port isolation technique where ports can be allowed to communicate with select predefined uplinks and not with other interfaces that belong to the same VLAN.
- **NetFlow:** Sends information on flows forwarded by the vDS. It is based on *tuples* - which contains source and destination IP addresses, transport protocol source and destination ports, among other data - that are received and processed by NetFlow collectors.
- **Port Mirroring:** Enables a copy of Ethernet frames to be directed to a local virtual machine or IP-connected traffic analyzer.

Defining Network Virtualization

The list of virtualization technologies in networking certainly does not end with VLANs, VRRP, VRFs, and virtual switching. In truth, it encompasses other device features such as:

- **Link aggregation**: Transforms two or more Ethernet interfaces into a single virtual port; defined by the IEEE 802.3ad standard
- **Device context:** Creates multiple virtual network devices that can be managed independently on the same switch, router, firewall, or server load balancer
- **Multi-chassis link aggregation:** Enables the aggregation of Ethernet links from one device connected to a pair of different Ethernet switches

A closer examination of these virtualization techniques shows that they share a recurring motif: they are focused on developing emulations of *network devices* (e.g., Ethernet switches, IP routers) or *a part of these devices* (e.g., switch port, router interface). However, these technologies do not create a complete network. And as a matter of fact, network designers are accustomed to adopting various multiple virtualization technologies to fulfill the connectivity requirements of applications and users.

In hindsight, because they were part of a long-term evolution, these virtualization features have never constituted a *virtualization platform* for networking in the same way that vSphere did for computing. For this reason, they have never radically reduced the number of operational procedures or established a framework for solution development.

Resultantly, networking remained a fairly complex subject until the 2010s. This was especially problematic in critical environments such as data centers, where the intricacy of network designs brought challenges such as provisioning time, security risks, site isolation, manual operations, and resource silos.

Because of their wider perspective, data center architects from large data centers could not avoid noticing that other infrastructure services that were also tightly coupled with hardware solutions - such as servers and storage arrays - similarly generated an excessive number of operational tasks that delayed application deployment and technology evolution.

Figure 1-9 hints at some of the challenges data centers face when deploying strictly hardware-based infrastructure.

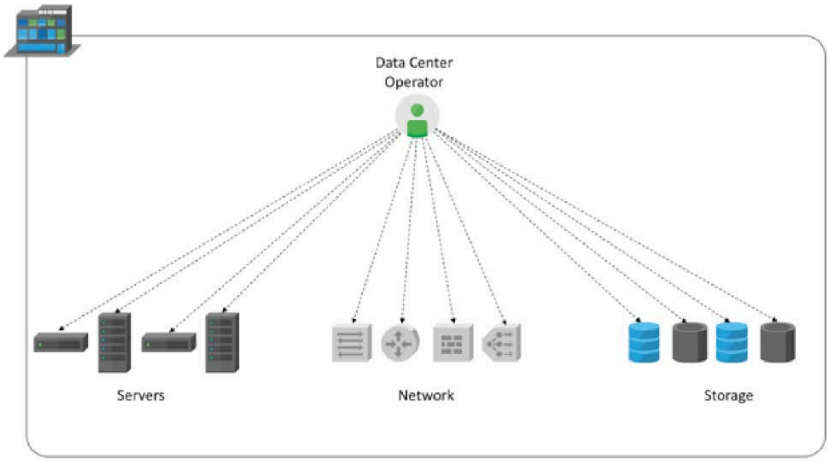


Figure 1-9 Data Center Based on Hardware Solutions

In Figure 1-9, the primary infrastructure services - processing, networking, and data storage - are dependent on the deployed features from specific hardware devices. Therefore, innovation must rely on the evolution of these devices or on a hardware "rip-and-replace" migration. Additionally, as the data center scales, new devices are needed and a greater number of operations must be executed to provision and manage the data center infrastructure. In practice, device segmentation technologies such as VLANs and VRFs may further aggravate this situation.

If different devices are acquired, additional skills must be learned and new processes must be created. In a worst case scenario, a complete data center redesign may be required. Many hardware-based features can only support a single data center site because of compatibility issues with the solutions deployed at other sites and service providers.

To address these challenges, mega-scale data centers from Internet and social network providers decided to adopt an architectural strategy called *Software Defined Data Center (SDDC)* to reduce complexity, save costs, and improve resource utilization. This IT strategy preaches that infrastructure services can be entirely abstracted as opposed of being tied to hardware.

Bringing SDDC principles to enterprise and service provider organizations with its SDDC architecture, VMware proposes that a *fully virtualized* data center can offer flexibility and dynamism not only in terms of computing with vSphere, but also in networking with NSX, storage with VMware vSAN™, and operations with the VMware vRealize® Suite.

Figure 1-10 outlines a scenario of VMware SDDC implementation.

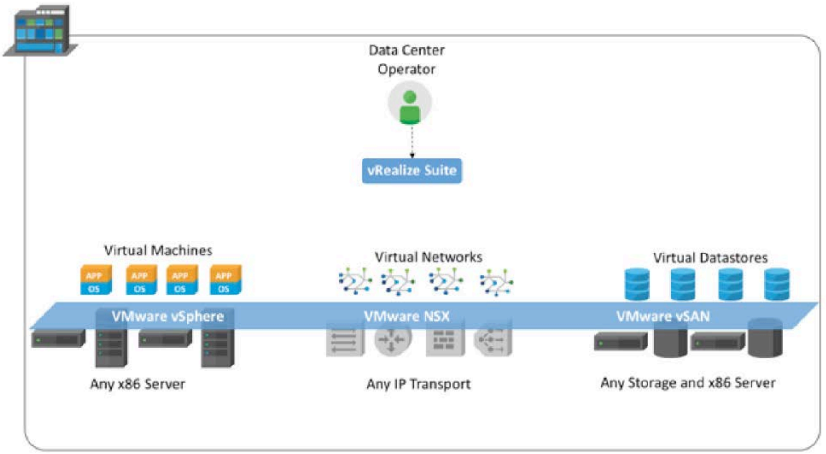


Figure 1-10 Software-Defined Data Center

As Figure 1-10 illustrates, a data center deploying VMware SDDC is independent from the previously acquired hardware. This frees data center architects to use solutions from multiple vendors simultaneously. In such scenario, the data center infrastructure becomes a geographically agnostic *shared pool* of resources that can be sliced up automatically.

VMware SDDC also drastically reduces the number of operation tasks as each virtualization platform - vSphere, NSX, and vSAN - entirely abstracts the number of distinct server, network, and storage solutions. Such a situation brings agility and flexibility to a data center infrastructure, facilitating automation, management, monitoring, governance, and business insight. This is especially evident when these functions are provided by the vRealize Suite, which is fully integrated with all three VMware virtualization platforms.

As an important outcome for organizations going through a digital transformation, VMware SDDC shifts personnel resources toward innovations and business growth while helping data center architects deliver secure, elastic and resilient infrastructure services.

To effectively support the SDDC objectives, NSX enables the creation of networks that are entirely decoupled from physical devices and can fully service modern applications. NSX provides *network virtualization,* which can be succinctly defined as the creation of complete virtual networks that are entirely autonomous from the data center physical infrastructure.

Figure 1-11 represents such endeavor.

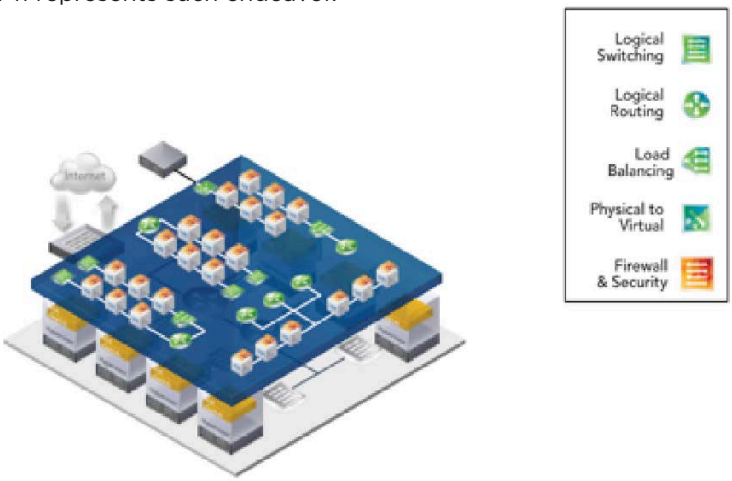


Figure 1-11 VMware NSX Network Virtualization

Figure 1-11 demonstrates that NSX can deploy multiple virtual networks, including services such as logical switching, logical routing, load balancing, firewalling, and other security features. These virtual networks can be provisioned, changed, and decommissioned as easily as virtual machines. They must be guaranteed full connectivity with all devices that are connected to the physical network - which includes the Internet.

Because NSX is a network virtualization platform, it greatly expedites the creation of new features, integration of different products, and development of innovative use cases.

The Data Center Network as a Service Provider Backbone

The network virtualization architecture proposed by NSX is not necessarily based on disruptive network redesigns or radical new configuration procedures. In fact, the reuse of network devices and concepts are part of the large majority of NSX rollouts.

VMware brings to data center networking some of the conceptual virtualization characteristics from service providers *backbones*, which can be defined as networks that tie together diverse networks.

The most common service provider backbones are based on MPLS. Created as a reaction to proprietary and hard-to-manage Frame Relay and Asynchronous Transfer Mode (ATM) backbones, MPLS is based on special routers that forward IP packets based on *labels* inserted between their L2 and L3 headers. For this reason, MPLS is sometimes referred as a "layer 2.5" technology. With this approach, MPLS provides the potential for creation of a myriad of services over through a flexible packet structure.

Because multiple labels can be stacked onto a single MPLS packet, a single data unit can store a variable amount of metadata. Consequently, a coordinated group of labels permits the implementation of several types of VPNs, including *layer 3 virtual private networks (L3VPNs).*

The L3VPN service allows an MPLS network to connect sites from different organizations in distinct VPNs. These VPNs are completely independent in terms of IP addressing and dynamic routing protocols, among other parameters.

Figure 1-12 displays the outcomes and basic architecture of an MPLS backbone providing layer 3 VPNs for three different organizations - Org1, Org2, and Org3.

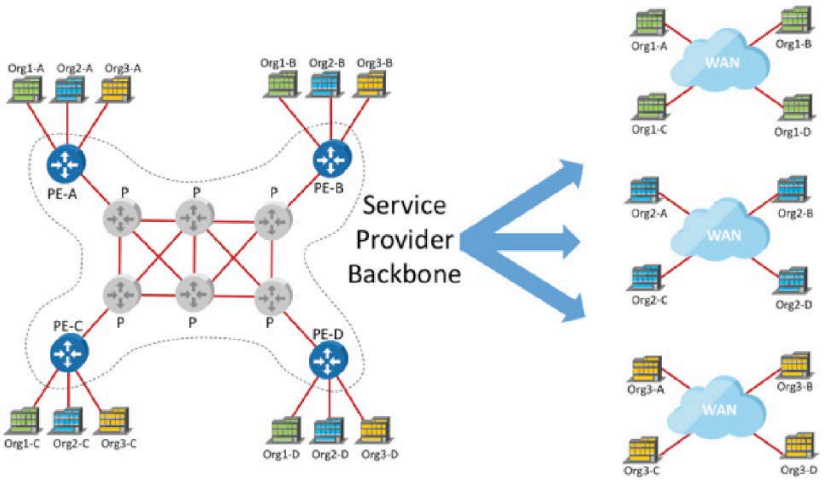


Figure 1-12 Layer 3 VPNs in an MPLS Backbone

In Figure 1-12, an MPLS backbone instantiates a separate layer 3 VPN for each organization, each of which have branches in four different localities - A, B, C, and D. Within its own VPN, each organization has the impression of having a dedicated WAN.

To offer these virtualized WANs, the set of specialized routers that comprise an MPLS backbone is divided into two basic types of devices:

- **Provider (P):** Routers focused on *label switching.* This involves receiving a packet with a label stack on an interface, reading the top label on the stack, checking its label routing table, changing the label, and sending the packet to an outgoing interface. These devices have a very simple configuration - basic IP dynamic routing for route discovery and MPLS enablement. Their main objective in the network is to provide *performance* and *reliability* to the backbone.

- **Provider Edge (PE):** Routers that are *directly connected* to customer edge (CE) devices located inside the remote branches. They deploy a separate VRF for each VPN to receive original packets from CEs, and attach labels to these packets in order to *signal* VPN parameters such as VPN identification and preferred backbone route.

When compared to other backbone technologies such as Frame Relay and ATM, MPLS is considered more flexible because P and PE routers do not have to share the same vendor or platform. They can also form any kind of physical topology to support distinct traffic interests for VPN customers. The creation of a VPN requires only configuration on the PE routers, contrary to Frame Relay and ATM, which required fairly complex configuration of virtual circuits in all network devices.

Additionally, traffic between branches can be exchanged through the shortest path in the backbone, not requiring any steering to a hub site which was common with VPNs based on Frame Relay and ATM.

VMware NSX can deploy virtual networks in a data center through a similar composition, which is represented in Figure 1-13.

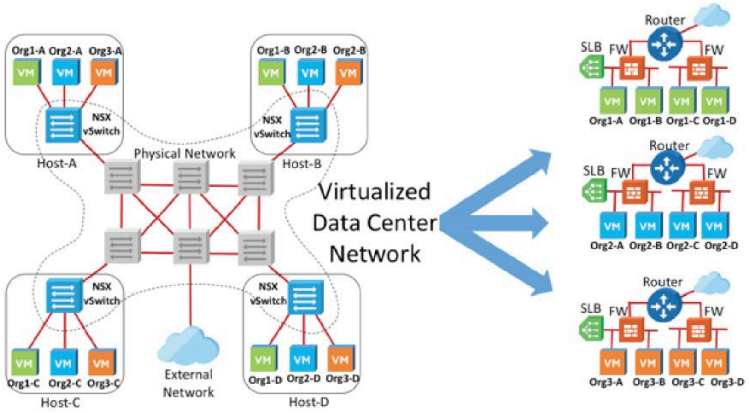


Figure 1-13 Network Virtualization with VMware NSX

As the Figure 1-13 explains, NSX creates virtual networks with various services for different set of virtual machines. Similar to service provider backbones, this network virtualization also depends on two types network devices:

- **Physical Switches:** These devices comprise the physical network. Much like provider routers, they are focused on packet switching with *performance* and *reliability*, and should not depend of convoluted configuration. In the case of NSX, the physical network only requires *IP unicast routing*. In general, MPLS-enabled devices are considered to too costly and complex for data centers network operations.

- **NSX vSwitch:** *Directly connected* to virtual machines from different organizations and applications, these virtual devices are focused on offering data center networking services such as layer 2 connectivity, routing, and firewall among many others. To correctly direct packets within the physical network, NSX vSwitches can leverage overlay protocols such as Virtual eXtensible Local Area Network (VXLAN).

Note: Chapter 2, "Rethinking Network Constructs", will explain VXLAN in detail. For now, this protocol can be understood as an encapsulation of Ethernet frames within IP packets.

There are many advantages of building virtual networks through such an architecture. NSX does not mandate specific devices or topologies from the physical network. NSX has freed network designers to adapt layer 3 networks according the data center traffic interest rather than vendor-specific limitations. This is further demonstrated in Chapter 6, "One Network".

After an initial simple configuration on each physical switch, all configurations are carried out through NSX, which will always optimize the path for traffic between VMs due to its distributed nature.

Different from service provider backbones, virtualized data center networks are built to support workload mobility enabled by virtualization features such as vMotion. Consequently, a network virtualization platform such as NSX must recognize the main network characteristics of VMs to provide consistent network services.

The next section will explain how it is designed to fulfill these particularities.

Note: VMware NSX can be considered a *Software-Defined Networking* (SDN) solution. However, because this acronym currently aggregates such a great number of disparate technologies, it will not be used for the rest of this book to avoid unnecessary confusion with solutions that are not focused on *network virtualization*.

VMware NSX Architecture

With an understanding of *what* is expected from a network virtualization platform, it is time to start satisfying curiosity about *how* NSX is internally architected.

For a better understanding, the next sections will delve into the NSX components according to a well-known network function classification called *network planes* - which include management, control, and data planes.

Figure 1-14 depicts the main NSX components and the network plane each belongs to.

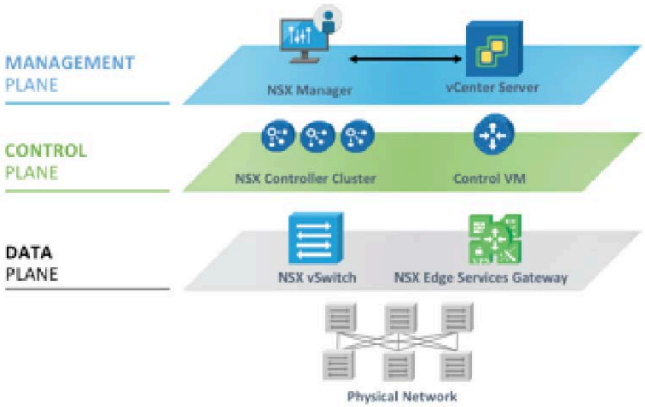


Figure 1-14 VMware NSX Architecture

Management Plane

Components in the *management plane* are responsible for the interaction of a network solution with external parties that may read or modify the behavior of elements that belong to other network planes. In the case of NSX, the VMware NSX® Manager™ performs the management plane role either through its graphical user interface (GUI) or application programming interface (API).

NSX Manager provides a single point of configuration for NSX as it is executed on a single specialized virtual machine. In the case of VMware NSX® for vSphere®, NSX Manager has a one-to-one mapping with a vCenter Server via a vSphere Web Client plugin.

This plugin enables the configuration via the vSphere Web Client, as Figure 1-15 highlights.

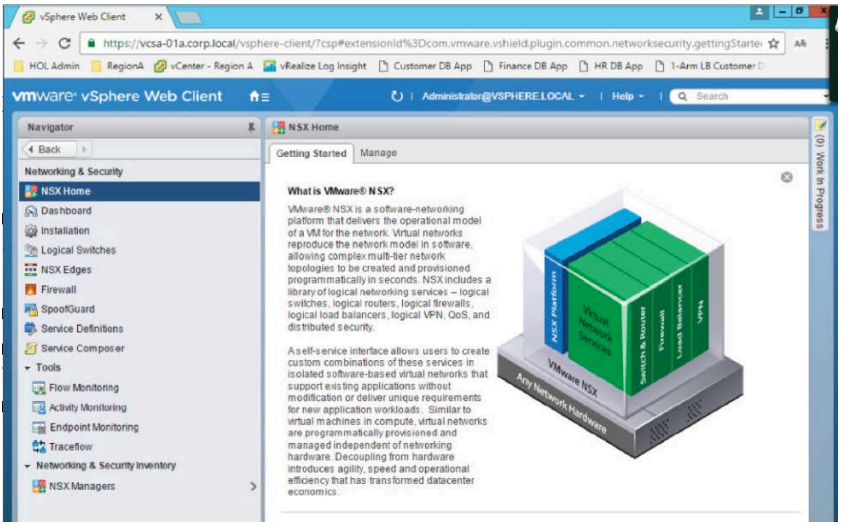


Figure 1-15 VMware NSX on vSphere Web Client

NSX installation heavily relies on NSX Manager, which is deployed as an Open Virtualization Appliance (OVA) using parameters such as IP addressing, vCenter Server, Domain Name System (DNS), and Network Time Protocol (NTP) among others. After its deployment and correct association to vCenter, NSX Manager coordinates the installation of the VMware NSX® Controller™ and the vSphere ESXi kernel module on each host for clusters that will run NSX services.

After NSX Manager configures the components of other network planes, they do not depend on the working state of the former to carry out their functions. For this reason, NSX deployments only require one virtual machine deploying NSX Manager. In case of a major failure, this instance can be quickly recovered - without traffic disruption - via a new NSX Manager VM and an imported configuration.

Control Plane

The *control plane* concerns the part of a network device or solution that defines how traffic forwarding will be carried out. In the case of NSX, this plane is materialized in the **NSX Controller cluster**, whose main functions are:

- To manage all logical networks and runtime state in an NSX deployment
- To collect VM network information (e.g., MAC and IP addresses) from a host and distribute to other hosts to optimize traffic exchange
- To reliably store all this information in the case of a failure of an NSX Controller that is part of the cluster

The NSX Controllers are virtual appliances provisioned by the NSX Manager during an NSX installation. Because they are exclusively a control plane element, they never sit in the data path.

To distribute processing load among its members, the NSX Controller cluster uses the concept of *slicing*, where all network virtualization objects are distributed across the nodes at least twice to guarantee recovery after a Controller node failure.

The minimum and maximum number of NSX Controller nodes per cluster is three. To ensure a higher availability of the cluster, they should be distributed across different hosts.

Note: An anti-affinity rule can be created on VMware vSphere® Distributed Resource Scheduler™ to avoid having two or more nodes automatically migrated to a single host.

The NSX Controller cluster communicates with the NSX Manager via a secure API. It also relies on a Secure Socket Layer (SSL) connection to communicate with a vSphere ESXi component called *user world agent* (uwa). The UWA gathers information and executes functions according to the cluster instructions.

Note: Each member of the NSX Controller cluster supports a CLI interface for troubleshooting procedures.

Another NSX control plane component is the **Control VM**, which deploys dynamic routing on behalf of a kernel-based data plane element called *distributed logical router* (DLR). This concept will be explained in more detail in Chapter 2.

Data Plane

Also known as forwarding plane, the *data plane* handles all traffic that traverses two or more interfaces of a network device or solution. In NSX for vSphere, two different elements represent this network plane:

- **NSX vSwitch:** Executes networking functions such as encapsulation, routing, and firewall in the hypervisor with line rate performance. It is comprised by the combination of vDS and hypervisor kernel modules configured in each host by the NSX Manager.
- **NSX Edge Services Gateway:** A specialized virtual appliance dedicated to the communication between the physical and virtual networks created within NSX. It offers advanced services such as dynamic routing, network address translation, firewall, load balancing, and virtual private networks among other features.

How exactly these elements are combined to invoke virtual networks is the entire subject of the next chapter, so hang on tight and read on!

What Did This Chapter Achieve?

This chapter introduced and explored the following concepts:

- **Formal definition of virtualization:** In the context of IT, virtualization can be defined as the faithful reproduction in software of an IT resource, delivering to its consumers benefits that were unavailable in its physical form.

- **Server Virtualization Revolution:** A different level of virtualization was achieved with server virtualization platforms such as VMware vSphere. Besides emulating physical servers, vSphere also enables advanced features such as VM high availability and vMotion.

- **Virtualization Techniques in Networking:** Virtualization is not a ew concept for networking, including techniques such as VLANs, VRRP, and VRFs.

- **Network Virtualization Definition:** Rather than simply emulating network devices, NSX offers the creation of complete virtual networks with a full set of networking services, such as layer 2 communication, routing, firewall, load balancing, and virtual private networks. As a virtualization platform, NSX has fomented new features and use cases that were not part of its original design.

- **VMware NSX Architecture:** NSX has components that work in each one of the network planes, which include NSX Manager (i.e., management plane), NSX Controller cluster and control VM (i.e., control plane), and NSX vSwitch and NSX Edge Services Gateway (i.e., data plane).

Rethinking Network Constructs

> *"I can't understand why people are frightened of new ideas. I'm frightened of the old ones."*
> *John Cage*

Although it is sometimes hard to accept, the large majority of network concepts applied to modern data centers are already in their forties -the ideal period for a midlife crisis. Protocols such as *Ethernet* - developed between 1973 and 1974 at Xerox Palo Alto Research Center - and the ubiquitous *Internet Protocol* - a product of the ARPANET efforts in the 1969 - are still very much the basis for data communication in such environments.

The fact that the same network primitives from these protocols are still being used today is a remarkable tribute to the elegance of their original design. Nonetheless, some of their aspects are naturally expected to become anachronistic due to seismic shifts in data center technologies. More specifically, server virtualization has clearly challenged data center networks in in the last decade.

This chapter discusses some of these challenges and also how NSX virtual network constructs such as distributed logical switches, distributed logical routing, and edge services can provide an elegant way to address these issues without a far-reaching change in operational terms.

Layer 2 Connectivity

Every IT professional with basic networking skills should leverage the *Open System Interconnect* (OSI) model as a firm backdrop for network design and operations. Published in 1984 by the International Standards Organization (ISO), this framework maps network functions onto seven layers to facilitate protocol development. Although this model was never systematically implemented, it serves as a masterful template for all network-related conversations and projects.

Figure 2-1 probably represents a quick walk down memory lane for anyone reading this book.

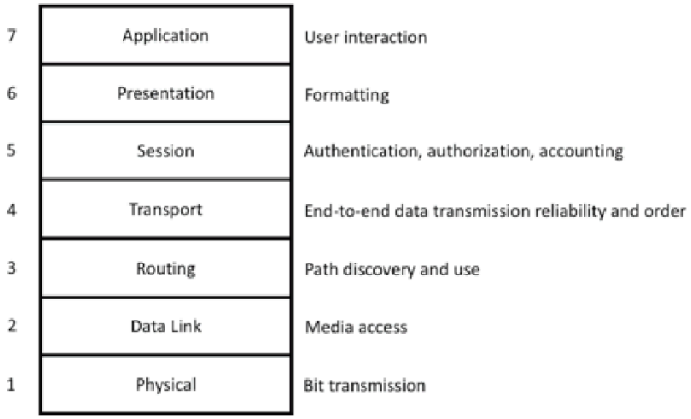


Figure 2.1 The OSI Model

In Figure 2-1, all seven layers are cited and their main objectives are briefly described to provide context on topics discussed in the next sections.

The *Data Link layer* is responsible for the communication between members of the same network; more specifically, neighbors that can exchange frames directly among themselves. In this context, a *frame* corresponds to a layer 2 data unit.

The most prevalent layer 2 protocol in data center networks is Ethernet. Due to its simplicity and openness, Ethernet has won all popularity contests over the last two decades against other technologies such as Token Ring and ATM. And because every server - physical or virtual - is expected to have an Ethernet *network interface controller* (NIC), the protocol has strongly defined how data center networks are designed and built.

Provisioning VLANs

As one of the many virtualization technologies discussed in Chapter 1, "What is Network Virtualization?", a VLAN is one of the most important network constructs provisioned in data center environments.

A new VLAN is commonly needed for one the following reasons:

- **IP address depletion:** It is a fairly common design for a VLAN to host a single IP subnet. When the number of available IP addresses in such a subnet are consumed, network operators are often forced to create a new VLAN to accommodate new hosts in a new subnet.
- **New application environment:** When a new application is deployed in a data center, network designers may prefer to dedicate a new VLAN or VLANs to the servers that are part of this application to exercise more control over such resources.
- **Security-based segmentation:** VLANs are commonly used to assert isolation or control over traffic between applications or application components in a shared data center network.

Note: Chapter 3, "Virtualization as a Security Control", will explore in greater detail the challenges involved with the use of VLANs for security-based segmentation.

Whatever the situation justifying their creation, Figure 2-2 diagrams the most common way VLANs are provisioned in a data center networks.

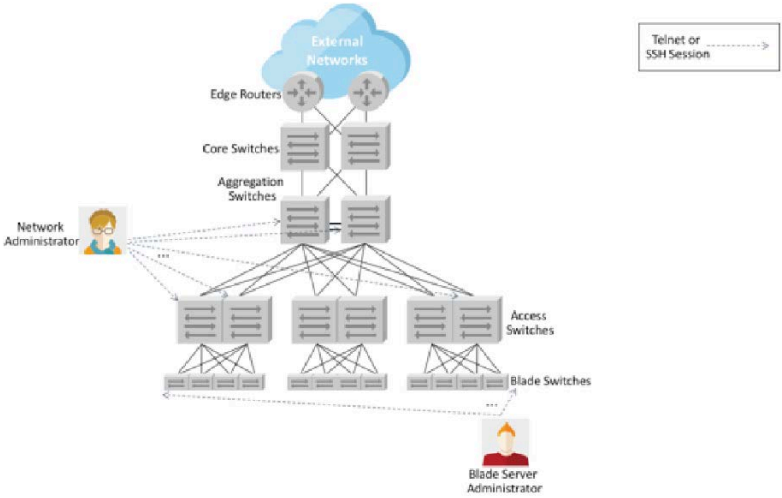


Figure 2-2 VLAN Provisioning

As Figure 2-2 shows, a VLAN is normally provisioned through a Telnet or Secure Shell (SSH) session to the CLI of each network device that may receive traffic on that broadcast domain. This figure shows a classic topology with core, aggregation, access switch pairs as well as an common additional layer of network devices installed inside blade servers. In such scenarios, all switches usually deploy VLANs except for the core switches that provide routing between different aggregation switch pairs and the external networks accessible via specialized edge routers.

Note: Chapter 6, "One Network" will discuss this topology in more detail along with the evolution of data center network architectures.

Example 2-1 generically represents what a network operator types during each CLI session to a switch.

Example 2-1 New VLAN Creation

```
! Entering configuration mode after a successful login
Switch# configuration terminal
! Creating the VLAN 100 on the switch
Switch(config)# vlan 100
Switch(config-vlan)# vlan name WEB
Switch(config-vlan)# exit
! Adding the VLAN to the switch interfaces that need to deploy
VLAN 100
Switch(config)# interface range Ethernet1/1-48,Ethernet2/1-4
Switch(config-int)# switchport trunk vlan 100 add
! Exiting configuration mode
Switch(config-int)# exit
Switch(config)# exit
Switch#
```

Considering that Ethernet switches are the most numerous network devices in data center environments today, executing the configuration described in Example 2-1 multiple times can be an extremely tiresome and error-prone process. A disastrous situation may unfold if the word **add** is missed when an **interface range** command is executed. This simple omission may erase all other VLANs in each of these trunks!

Because of such risk, many network operators have abandoned CLI-based automation shortcuts for provisioning. Seasoned readers may remember switch functionalities such Cisco VTP (VLAN Trunk Protocol) and Multiple VLAN Registration Protocol (MVRP), which intended to propagate VLAN configuration from a *server* switch toward all other *client* switches within a LAN. As misconfigurations and outages happened with the use of such protocols, most network vendors recommended that data center switches are put on *transparent* mode - where these devices do not participate on any configuration propagation provided by these protocols.

A question seldom heard from many non-networking professionals is: "*Why not provision all 4,094 VLANs in every port and save all provisioning hurdles*?" While a valid question, it overlooks some factors such as undesirable broadcast traffic and scalability limits such as the number of supported VLANs per network device and spanning-tree protocol active ports.

Other readers may argue that network management tools can accelerate network provisioning. Unfortunately, such software solutions are usually developed with a single network vendor or platform in mind, which can greatly restrict its use in heterogeneous data center networks. On the other hand, multi-vendor large-scale network management tools may provide a limited number of provisioning features because of their broad scope of managed devices.

As few enterprise data centers rely on such network provisioning techniques, a large amount of time is usually taken to deploy new VLANs within a data center network. Because such operation pose the risk of major impact, most companies only allow VLAN configuration during planned *maintenance windows* - usually occurring on weekends or select days of each month where there is minimal application activity.

Introducing Logical Switches

It seems almost unfair to compare the grievances that network operators go through with the operations performed by the server virtualization team during a new application deployment. While weeks or even months are spent to deploy a set of VLANs along with other network constructs, a server virtualization operator can create all virtual machines a new application needs in seconds.

Although it may seem like pure magic, more astute readers certainly are aware that the agility server virtualization enjoys is explained by very palpable concepts:

- **Virtual machines are software constructs:** VMs are simply files that are processed by hypervisor software with control over physical hardware. Consequently, the creation of a virtual machine leverages the flexibility a file offers such as cloning and transfer between different computers.

- **Centralized management:** Serious server virtualization deployments use a central management tool such as vCenter Server to coordinate multiple operations on a set of hypervisors rather than per-host or per-VM tasks.

- **VM creation does not disrupt other deployed applications:** A new VM is a complete independent entity, which shares hardware with other VMs but does not create significant risk to their operations.

As detailed in Chapter 1, virtual networking has greatly evolved in terms of scale with the *vSphere Distributed Switch*, which made it possible to simultaneously create distributed port groups in multiple hosts. While

VLANs can be provisioned within such port groups, the layer 2-only nature of the vDS still requires the provisioning of the same VLANs in the physical infrastructure of the data center network.

Strictly speaking about layer 2 connectivity, NSX proposes the following alternatives for network designers:

1. *What if broadcast domains could be exclusively created in software without requiring any additional operations on physical devices?*

2. *What if such broadcast domains were provisioned through a centralized management tool?*

3. *What if the addition of these domains presented minimal risk to previously deployed layer 2 segments?*

Bringing reality to such inquiries, NSX deploys a network object called a *logical switch*, which connects virtual machines to the same broadcast domain. Distinctively from VLANs, a logical switch is created with a single operation on NSX Manager, as Figure 2-3 illustrates.

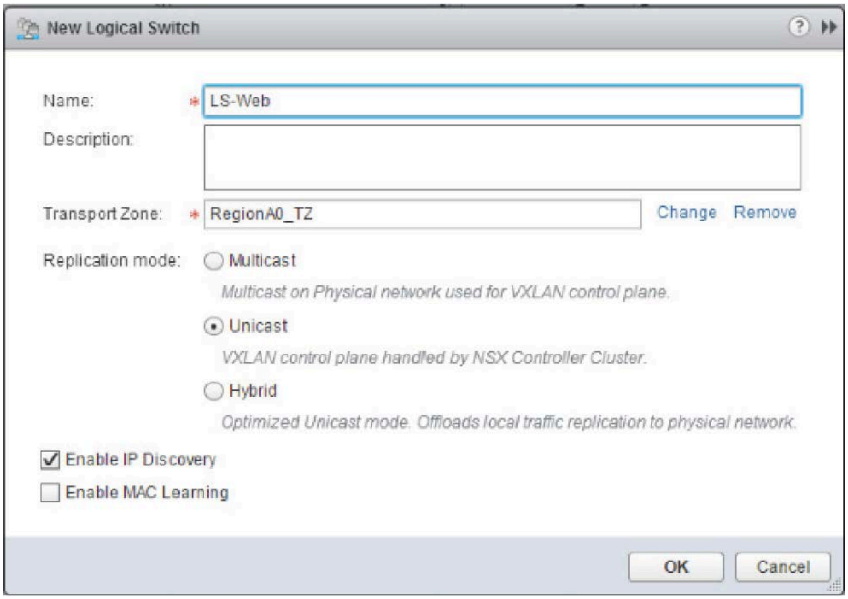


Figure 2-3 Creating a Logical Switch

Figure 2-3 depicts the parameters NSX requires to provision a logical switch named **LS-Web.** As soon as the **OK** button is pressed in this screen, NSX Manager instructs the NSX Controllers to program a new logical switch on all hosts that are part of *transport zone* called **RegionA0_TZ**. A transport zone can be formally defined as a group of virtualized hosts that can share a logical switch. This operation automatically provisions a new distributed port group to which vNICs of virtual machines can be attached.

From a connectivity perspective, logical switches do not behave too differently from VLAN-based distributed port groups for VMs running on the same host. In these scenarios, if two or more VMs are connected to **LS-Web** they will be able to exchange Ethernet frames.

Figure 2-4 explains how logical switches address the layer 2 communication between virtual machines located in different hosts.

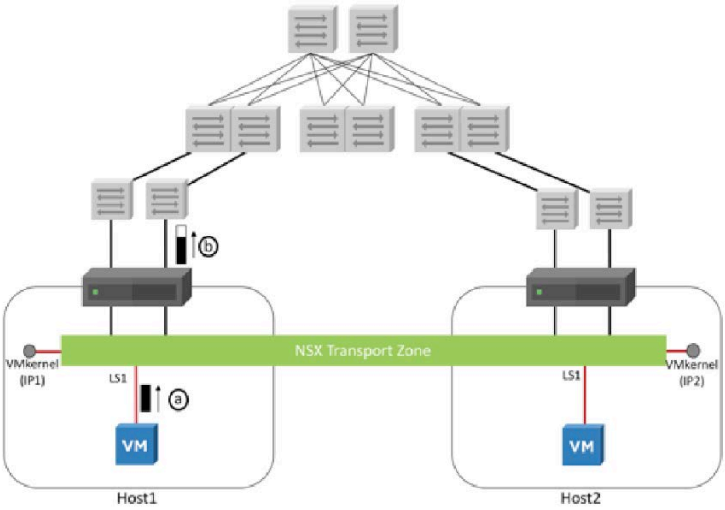


Figure 2-4 Virtual Machine Sending an Ethernet Frame on a Logical Switch

Figure 2-4 details the following actions:

a. A virtual machine running on Host1 and connected to logical switch **LS1** sends an Ethernet frame.

b. Host1 encapsulates such a frame into a special IP packet defined by an *overlay protocol*, such as Virtual eXtensible LAN (VXLAN). This packet has the IP1 – the Host1 VMkernel interface IP address - as its source address.

Figure 2-5 portrays the same frame being received by a virtual machine running on Host2.

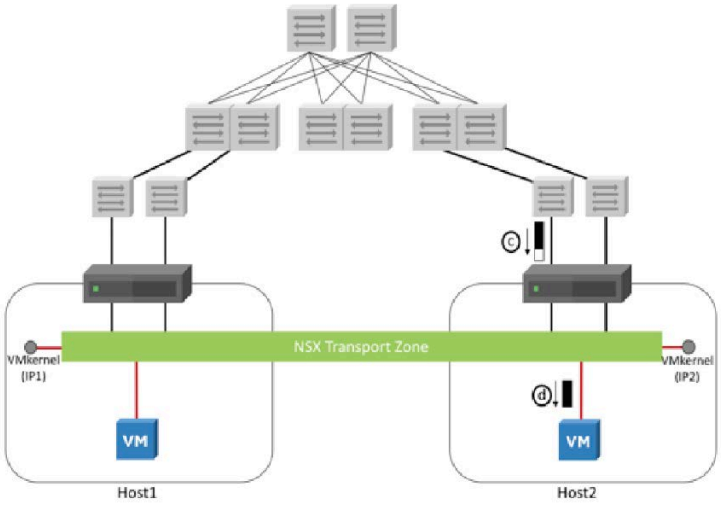


Figure 2-5 Virtual Machine Receiving an Ethernet Frame on a Logical Switch

Figure 2-5 delineates the following operations:

a. Because the destination IP address on the overlay protocol packet is IP2, this packet arrives at a VMkernel interface on Host2. The host then identifies the logical switch the packet belongs to **LS1** and decapsulates it.

b. Back to its original format, the Ethernet frame is finally delivered to the destination VM running on Host2.

As a result, virtual machines can easily exchange Ethernet frames through an NSX logical switch without requiring VLAN provisioning processes in the physical network, which is commonly called *underlay network* in such scenarios. Because overlay protocols exclusively rely on IP routing, physical network devices are configured only once for NSX deployments.

Although there are many other Ethernet-over-IP protocols, VXLAN is arguably the most used. Created in 2011 through a collaboration of VMware, Cisco, and other vendors, it was published as standard Request for Comment (RFC) 7348 by the Internet Engineering Task Force (IETF) in 2014.

Figure 2-6 describes the VXLAN encapsulation header.

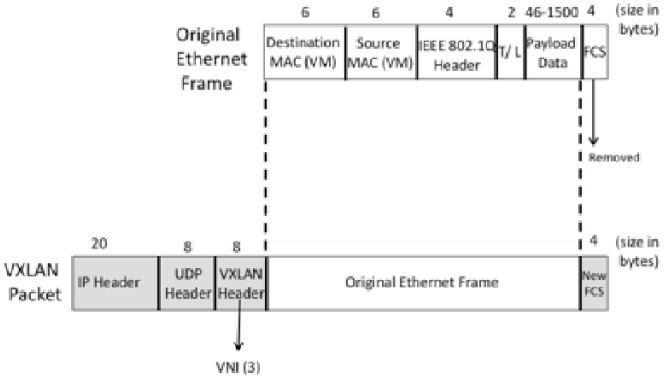


Figure 2-6 VXLAN Encapsulation Header

As Figure 2-6 clarifies, VXLAN encapsulates Ethernet frames inside UDP datagrams, which in theory require IP routing to be delivered between *VXLAN Tunnel End Point*s (VTEPs). In the case of Figures 2-4 and 2-5, Host1 and Host2 are software VTEPs that use their VMkernel addresses - IP1 and IP2, respectively - to encapsulate and decapsulate VXLAN packets.

A VXLAN control plane establishes how each VTEP discovers the MAC addresses associated with other VTEPs to correctly forward traffic. Although RFC 7348 states IP multicast routing as its method of choice for such discovery process, the network industry has developed innovative control plane techniques for overlay protocols.

Because of its stark simplicity, the vast majority of NSX deployments use the NSX Controllers to perform control plane functions for overlay protocols. As these elements gather and distribute information to all VTEPs, only IP unicast routing is actually required from the underlay network. In addition, NSX can also deploy multicast-based VXLAN - per logical switch or transport zone - to allow smooth compatibility with existing network configurations already in place.

Note: Generic Network Virtualization Encapsulation (GENEVE) was created as a reaction to the lengthy discussion around the validity of VXLAN control plane options as well as the protocol's lack of flexibility to add metadata to its header. Actively led by VMware, Microsoft, and Intel, GENEVE defers the control plane design to each vendor implementation while focusing on data plane aspects and encouraging innovation on other planes. From a packet format perspective, GENEVE is similar to VXLAN, except for a variable field called "Options". This field was designed to support extensible metadata for various uses such

as telemetry, traffic classification, and service insertion. NSX for vSphere – also referred to as NSX-V - supports VXLAN while NSX-T supports GENEVE. Always refer to NSX documentation to verify the availability of VXLAN and GENEVE for the NSX version in use.

Overlay protocols such as VXLAN and GENEVE represent the culmination of a trend present since the creation of the Internet Protocol.

Since its beginnings, IP has been essentially used as a normalization protocol to easily provide data communication between hosts connected to heterogeneous layer 2 technologies such as Ethernet, Token Ring, Frame Relay, and many others. Its ubiquity also allowed IP to also become a transport backbone to interconnect geographically-bound systems through technologies such as:

- Voice-over-IP (VoIP)
- Video-over-IP (live and on-demand)
- Internet Small Computer System Interface (iSCSI)
- Fibre Channel over IP (FCIP)

Overlay protocols use the same principle of *extending* technologies such as Ethernet LANs, which are usually bounded by data centers facilities, campus areas, corporate buildings, or branch offices. For example, using logical switches, NSX can provide layer 2 connectivity to VMs with the ease of a single provisioning operation, regardless of their location on an IP unicast physical network.

Figure 2-7 demonstrates the final effect of a logical switch creation in an NSX deployment.

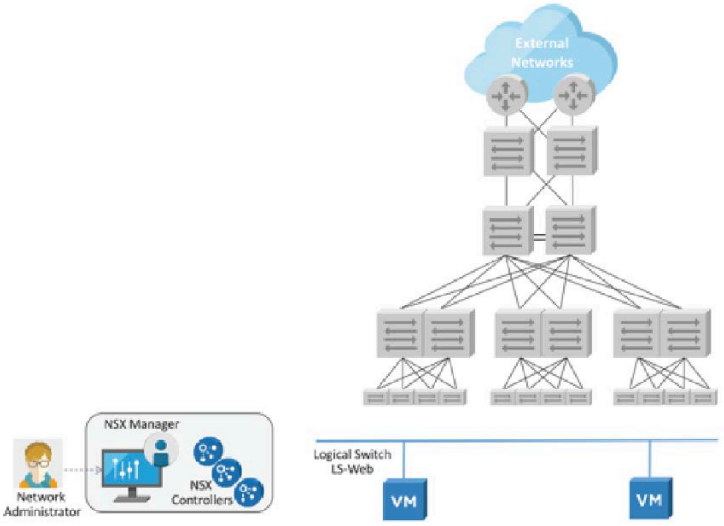


Figure 2-7 Logical Switch Creation

Note: As with many of its features, the use of logical switches is optional in an NSX deployment. Should they be used to simplify layer 2 connectivity provisioning, it is a requirement that the Maximum Transfer Unit (MTU) in the physical network is increased to accommodate the extra bytes added by the overlay protocol headers. A minimum 1600 bytes MTU for VXLAN and 2000 bytes MTU for GENEVE is recommended.

Layer 3 Connectivity

According to the OSI model presented in Figure 2-1, layer 3 is responsible for discovering at least one path between different layer 2 networks and using it for packet forwarding. Informally, both operations are summarized in one simple word: routing.

Particularly in data center networks, layer 3 devices provide communication between hosted servers and external clients - *north-south traffic* - and also among such servers – *east-west traffic*.

Whereas there are multiples approaches to provide L3 connectivity in a data center network, routing functions have been traditionally implemented in a centralized way. Only a few devices in such environments provide communication of packets between different IP subnets, which are usually associated with VLANs in a 1-to-1 relationship.

Figure 2-8 introduces such a classic approach in data center networks.

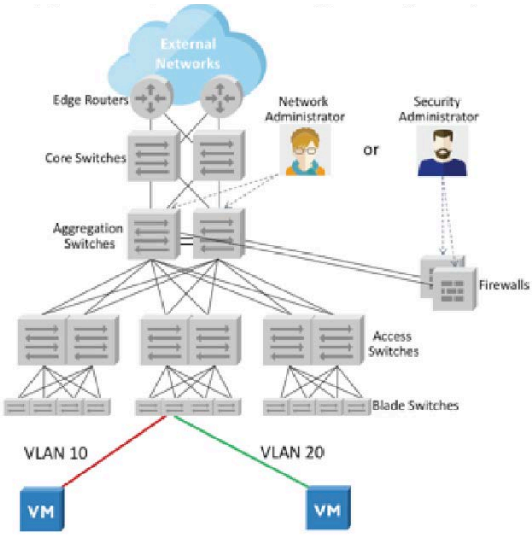


Figure 2-8 Traditional Centralized Layer 3 in Data Center Networks

As detailed in Figure 2-8, L3 routing services are usually configured on aggregation switches or on firewalls. These devices are expected to handle both north-south and east-west traffic between servers that belong to different subnets, such as the VMs located in VLANs 10 and 20.

Example 2-2 shows the usual configuration of L3 routing services in each switch of an aggregation pair of devices. In this instance, VLANs 10 and 20 were already created through the commands explained in Example 2-1.

Example 2-2 IP Routing Configuration on an Aggregation Switch

```
Switch# configure terminal
Switch(config)# feature interface vlan
Switch(config)# feature vrrp
Switch(config)# interface vlan 10
Switch(config-if)# ip address 10.1.1.1/24
Switch(config-if)# no shutdown
Switch(config-if)# vrrp 10
Switch(config-if-vrrp)# address 10.1.1.254
Switch(config-if-vrrp)# no shutdown
Switch(config)# interface vlan 20
Switch(config-if)# ip address 20.1.1.1/24
Switch(config-if)# no shutdown
Switch(config-if)# vrrp 20
Switch(config-if-vrrp)# address 20.1.1.254
Switch(config-if-vrrp)# no shutdown
```

While Example 2-2 may represent a relative relief in terms of operational effort as there are fewer devices to configure compared to VLAN provisioning, the centralization of IP routing services produces specific side effects for east-west L3 traffic:

- **Additional hops:** More network hops are necessary to provide communication between servers - virtual or physical - that may be running on the same host or on different hosts connected to the same physical switch.
- **Uplink higher utilization:** Both north-south and east-west traffic may compete for such resources, which are usually oversubscribed, meaning that they support less than the maximum potential bandwidth that can be generated by the connected servers.
- **Overload of stateful devices:** Where a firewall is providing L3 services instead of aggregation switches, it may become overwhelmed with the sheer volume of east-west traffic between different IP subnets.

The centralization of L3 routing services is so ingrained in traditional designs of data center networks that these drawbacks are generally accepted as the sour taste of a medicine. In spite of this, joining the not-so-silent networking revolution during the early 2010s, VMware NSX offers a different approach to these basic services.

Introducing Distributed Logical Routing

As explained in Chapter 1, virtual switching has significantly evolved since the creation of the vSS in 2003, and furthermore in 2009 with the introduction of the vDS. Both virtual devices set off ripples in the still young server virtualization market and inspired similar deployments in all other x86-based hypervisor platforms.

These virtual networking devices only offered L2 services, which included VLAN encapsulation to physical switches via the IEEE 802.1q standard and frame forwarding belonging to the same VLAN inside the hypervisor. Resultantly, *switched* east-west traffic was distributed among the hosts deploying these virtual switches, so whenever VMs were hosted in the same physical server, L2 communication did not depend on the physical network.

But here is an interesting inquiry: *What if these virtual network devices could do the same with L3 functions? What if they could route traffic between VMs in two different subnets?*

Within NSX, such intentions are materialized with a network construct called a *distributed logical router (DLR)*.

Figure 2-9 portrays two basic behaviors of the NSX DLR.

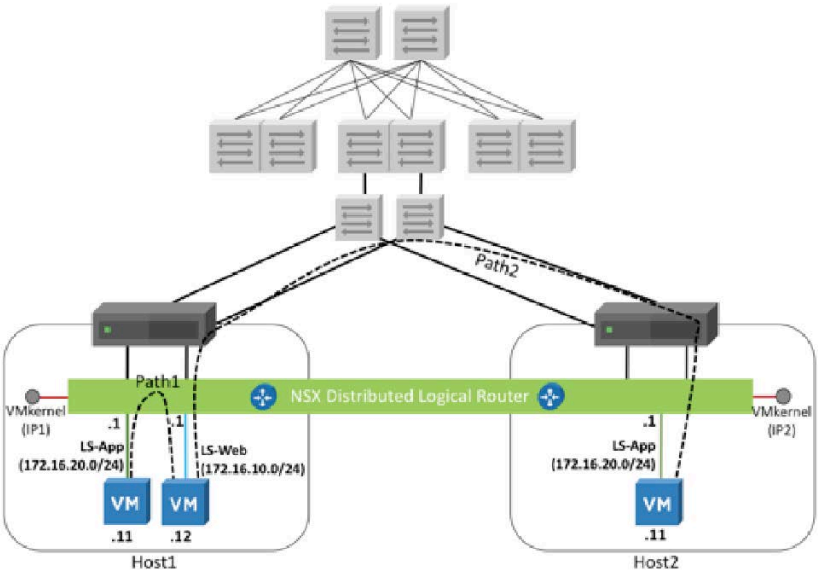


Figure 2-9 Distributed Logical Routing in Action

In Figure 2-9, virtual machines in different subnets can leverage the routing services on an NSX DLR, which offloads the aggregation switches or firewalls from this east-west communication.

Distributed logical routing works in tandem with logical switching, as Figure 2-9 exemplifies with two paths, which are creatively named *Path1* and *Path2*.

In Path1, two VMs are located in the same host - Host1. These VMs belong to different subnets - 172.16.10.0/24 and 172.16.20.0/24 - and logical switches - LS-Web and LS-App, respectively. The DLR enforces routing between them inside Host1 through default gateways - 172.16.10.1 and 172.16.20.1 respectively - that exist on all hosts deploying the DLR. Because this path only exists inside Host1, there is no overlay protocol encapsulation in this scenario.

In Path2, the intercommunicating VMs are located in two different hosts - Host1 and Host2. The DLR fulfils its routing functions in Host1, where the source VM is located through the distributed default gateway associated to LS-Web. After the DLR receives packets on its interface with IP address 172.16.10.1, the inter-VM communication is performed within LS-App, which is the logical switch where the destination VM is located.

Note: More security-oriented readers are probably concerned about traffic from two network segments being routed without the careful analysis of a firewall. This thought is examined in Chapter 3, where a VMware NSX feature called *distributed firewall* will be explained in more detail. The most common VMware NSX designs attest that both DLR and distributed firewall should work together to guarantee - and even augment - security in data centers using server virtualization on any scale.

Like logical switching, distributed logical routing is a kernel-level process. Data plane functions are integrated into the hypervisor code, optimally using its controlled hardware resources such as CPU, memory, storage, but with minimal overhead when compared to networking services that are based on virtual machines.

Such implementation decisions make a lot of sense when understanding that east-west traffic is responsible for a continually increasing majority of traffic inside traditional data centers[1].

For this reason, many NSX deployments deliver a sensible reduction of bandwidth on the uplinks as well as a decrease in latency between VMs.

[1] https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf

Using distributed logical routers, NSX can provide L3 connectivity to virtual machines running on multiple hosts with the provisioning of a single network construct. This also provides per-host scale out performance and centralized management via NSX Manager.

Figure 2-10 delineates the effect of adding another logical switch and a DLR to the evolving NSX virtual topology last represented in Figure 2-7.

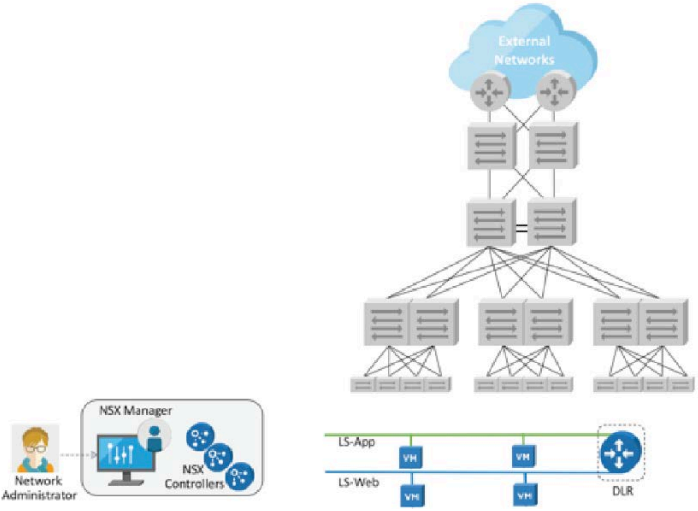


Figure 2-10 Virtual Topology with DLR Addition

Note: Depending on each project requirements, an NSX deployment may spawn multiple DLRs to ensure separation of network environments that include overlapping IP subnets, such as some multitenant scenarios. Please refer to appropriate VMware documentation to verify how many DLRs are supported by the NSX version in use.

Starting to Get Physical

The server virtualization revolution has dominated enterprise and service provider data centers since its explosion in the mid-2000s. But unsurprisingly, many of these environments still maintain physical workloads for assorted reasons. It is only fair to expect that such devices will also communicate with VMs connected to NSX logical switches.

VMware NSX provides connectivity with physical workloads through various methods. Among others, the DLR offers a feature called *Layer 2 Bridging* - even though this device was originally designed to provide optimized L3 services for VMs.

Figure 2-11 explains such functionality.

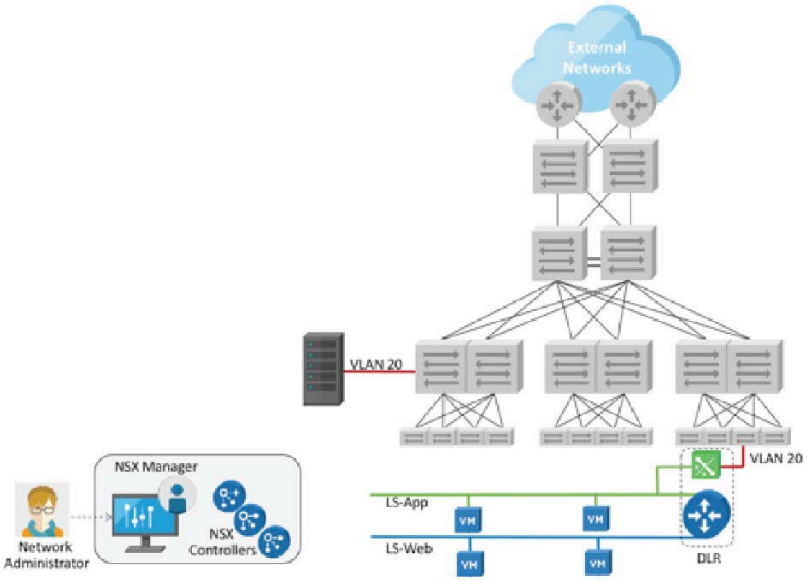


Figure 2-11 Layer 2 Bridging in Action

As Figure 2-11 introduces, layer 2 bridging allows workloads connected to a standard VLAN – VLAN 20 in the case of the figure - to directly exchange Ethernet frames with VMs connected to a logical switch (LS-App). This VLAN-logical switch binding is done within the DLR on one of the hosts that implements it. As a consequence, it is executed at the kernel level.

This feature can be further scaled out through the deployment of more L2 bridges, whose configuration is shown in Figure 2-12.

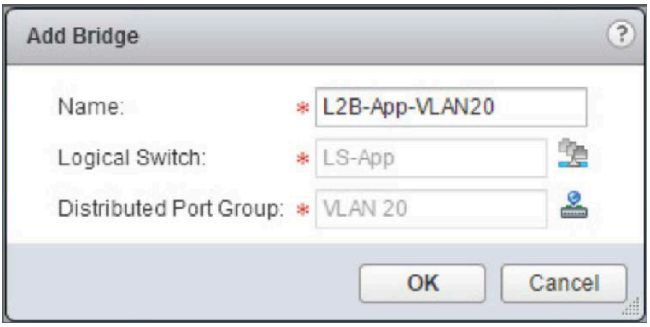


Figure 2-12 Configuring a Layer 2 Bridge in VMware NSX

Layer 2 bridging is commonly used in the following scenarios:

- **Ongoing server virtualization process:** When active L2 communication is expected during the transformation of physical servers into virtual machines without any IP address changes.
- **Virtual machine migration to VMware NSX:** Virtual machines are connected to standard or distributed port groups with a defined VLAN and will be connected to a logical switch.
- I**ntegration with networking services appliances:** When physical firewalls, intrusion prevention systems, server load balancers, and application delivery controllers deployed on physical appliances may require L2 connectivity with VMs connected to logical switches.

The majority of traffic between VMs and physical servers does not require direct L2 communication. For the sake of simplicity, routing is recommended for physical-virtual traffic, as the next section will address.

Living on the Edge

In terms of pure packet forwarding, logical switches and distributed logical routers are primarily designed to handle east-west traffic (i.e., communication between servers) as discussed on section "Starting to Get Physical".

VMware NSX for vSphere handles connectivity between devices connected to an NSX virtual network and external elements connected to the physical network - north-south traffic from NSX perspective - through a flexible component called *Edge Services Gateway* (ESG or NSX Edge).

The NSX Edge is a highly specialized virtual appliance designed to perform network operations such as:

- Edge Routing
- Network Address Translation (NAT)
- Server Load Balancing
- Edge Firewall
- IPsec Virtual Private Network (VPN)
- Secure Socket Layer Virtual Private Network (SSLVPN)
- Layer 2 Virtual Private Network (L2VPN)

The next sections discuss the first three functions from this list, while all VPN-related features will be discussed in Chapter 3. L2VPN will be addressed in Chapter 4, “Virtual Networks, Real Extensions”.

In addition to these data plane functions, the NSX Edge can also perform auxiliary network infrastructure operations that will be further discussed on section “Other Interesting Edge Features” in this chapter.

Note: Because NSX is a network and security platform based on software, its staggering development is a challenge to any writing. Be sure to refer to the latest NSX documentation to verify which features are actually deployed in any given version.

Edge Routing

In the large majority of NSX deployments, the ESG is the sole element provides *edge routing* between the virtual and physical environment. As an illustration, Figure 2-13 depicts a common virtual topology with this functionality deployed on an NSX Edge.

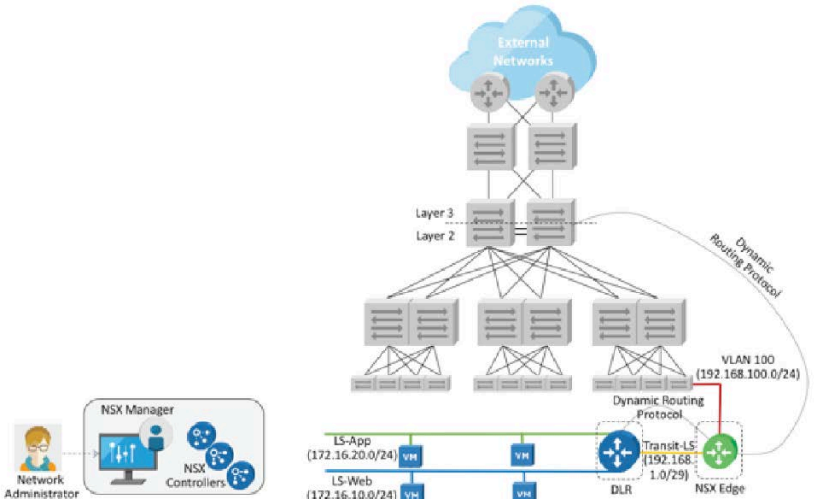


Figure 2-13 NSX Edge Routing Example

In this figure, the NSX Edge provides bidirectional L3 communication to logical switches connected to the DLR using a special logical switch called **Transit-LS**. This binds the DLR and the NSX Edge through a L2 connection.

An outbound IP packet from a virtual machine connected to **LS-Web** reaches the DLR and is routed via the transit logical switch toward the NSX Edge. The NSX Edge is connected to an upstream VLAN - 100, in the figure - representing the default route to the physical network.

On the reverse path, the NSX Edge routes an inbound packet received from VLAN 100 through Transit-LS so it can reach DLR-connected logical switches such as LS-Web and LS-App.

Note: An NSX Edge can have up to 10 vNICs connected to a mix of port groups – either standard or distributed - or logical switches. The NSX Edge supports trunk vNICs with associated sub-interfaces connected to individual VLANs. Always refer to the latest VMware documentation to verify the updated maximum configurations for the version in use.

Because both the DLR and the NSX Edge support standard dynamic routing protocols such as OSPF and BGP, **Transit-LS** is used to exchange IP subnet information between both devices. Similarly, the latter can deploy such protocols to advertise the IP subnet information received from the DLR to the physical network and vice-versa.

Note: If for some reason dynamic routing is not preferred, configuration of static routes on both virtual network devices and physical network can be performed whenever a new IP subnet created or deleted.

Both NSX Edge and DLR also support redistribution of routes into dynamic routing protocols. It is considered a good practice to redistribute all *connected* subnets to the dynamic routing protocol of choice to avoid reconfigurations if a new logical switch is connected to the DLR. With such arrangement, any new IP subnet is automatically imported into the routing protocol.

Example 2-3 explores the routing table of an NSX Edge in Figure 2-13 through a SSH session toward the Central CLI in NSX Manager.

Example 2-3 NSX Edge and DLR Routing Tables

```
Using username "admin".
admin@nsxmgr-01a.corp.local's password:
Listing all edges on this NSX Manager instance (commands includes the DLR)
nsxmgr-01a> show edge all
NOTE: CLI commands for Edge ServiceGateway (ESG) start with 'show edge'
      CLI commands for Distributed Logical Router (DLR) Control VM start
with 'show edge'
      CLI commands for Distributed Logical Router (DLR) start with 'show
logical-router'
      Edges with version >= 6.2 support Central CLI and are listed here
Legend:
Edge Size: Compact - C, Large - L, X-Large - X, Quad-Large - Q
Edge ID                  Name                              Size Version
Status
edge-3                   Perimeter-Gateway-01      C       6.2.3    GREEN
edge-6                   Distributed-Router-01     C       6.2.3    GREEN
Discovering 'Perimeter-Gateway-01' routing table
nsxmgr-01a> show edge edge-3 ip route
haIndex:

Codes: O - OSPF derived, I - IS-IS derived, B - BGP derived,
C - connected, S - static, L1 - IS-IS level-1, L2 - IS-IS level-2,
IA - OSPF inter area, E1 - OSPF external type 1, E2 - OSPF external type 2,
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

Total number of routes: 5

B                0.0.0.0/0                [20/0]   via 192.168.100.1
O        E2      172.16.10.0/24           [110/1]  via 192.168.5.2
O        E2      172.16.20.0/24           [110/1]  via 192.168.5.2
C                192.168.5.0/29           [0/0]    via 192.168.5.1
C                192.168.100.0/24         [0/0]    via 192.168.100.3
nsxmgr-01a>
```

Note: The DLR relies on a specialized virtual machine called *Control VM* to perform its dynamic routing protocol duties in NSX. Because it is exclusively a control plane entity, no traffic ever traverses this VM.

An NSX Edge can potentially run on any host from a server virtualization implementation as long as its upstream VLANs are reachable. Using Figure 2-13 as an example for such scenario, the upstream VLAN 100 should be configured on every physical and virtual switch to guarantee unrestricted mobility for the depicted NSX Edge.

On the other hand, as Chapter 6 will discuss in further detail, some physical network topologies do not support the implementation of the same VLAN on all access ports. In these cases, a separate cluster of hosts can be exclusively used to run NSX Edges, guaranteeing that all potential upstream VLANs are available for these virtual appliances. This special set of hosts is called an *Edge Cluster*.

NSX Edge High Availability

The question "*What happens if the NSX Edge fails?*" should be front of mind after reading the last section. This section explains how NSX provides high availability to NSX Edges.

Figure 2-14 introduces both NSX Edge high availability models.

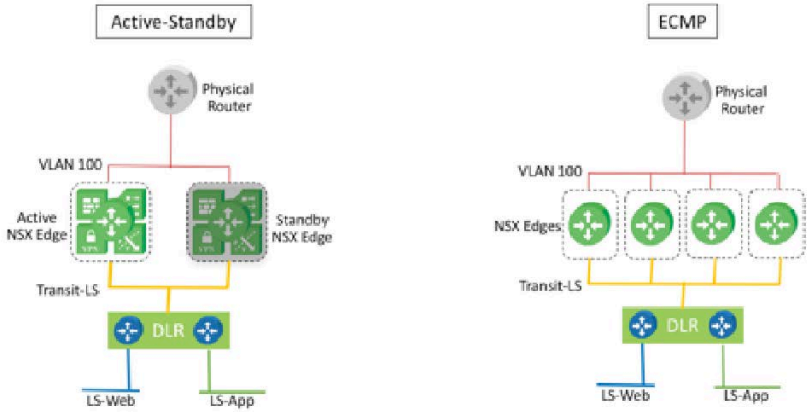


Figure 2-14 NSX Edge High Availability Models

On the left side of Figure 2-14, two NSX Edges are configured in *active-standby* mode, where an active NSX Edge is performing all actions while a standby NSX Edge is not executing any packet forwarding. The latter is eagerly waiting to become active in case any of the deployed functions on the active NSX Edge fail.

In order to communicate their health state to one another, the NSX Edges in an active-standby pair use a logical switch - which can be one that already exists, such as **Transit-LS** in the figure - to exchange "heartbeat" messages. It is possible to configure heartbeat parameters on an NSX Edge pair, such as dead declare time and vNIC.

The right side of Figure 2-14 introduces an alternative to the active-standby mode that is commonly known as *Equal Cost Multipathing* (ECMP). In this mode, up to eight NSX Edges with similar configuration are put side-by-side to route packets between the DLR and the physical network. In the case of one of the NSX Edges failing, packets will be automatically balanced to the surviving virtual devices.

Note: For ECMP mode to work, the DLR and external router must deploy Equal Cost Multipathing themselves to distribute packets fairly to all NSX Edges on a ECMP group.

As philosopher and Catholic saint Thomas Aquinas has accurately stated: *"Each choice is a renunciation"*. So what does one give up when choosing one NSX Edge availability model over the other? Table 2-1 summarizes the factors weighing on such dilemma.

Table 2-1 Comparison of NSX Edge Availability Models

Characteristic	Active-standby	ECMP
Scalability	NSX Edge size	NSX size and number
Available Services	Edge routing, NAT, Load Balancing, Edge Firewall, IPsec VPN, SSL VPN, L2VPN	Edge routing

As Table 2-1 shows, each model treats NSX Edge scalability and services distinctly. From a scalability perspective, and because the active-standby model relies on a single active NSX Edge, it can only grow its capacity through the choice of size of the NSX Edge HA pair.

Note: Four standard sizing templates are available: compact, large, X-large, and quad-large. Refer to VMware documentation to verify the performance and requirements for each NSX Edge size.

Besides sizing - which must be the same for all NSX Edges in a set - the ECMP model can also deploy up to eight NSX Edges. Because the ECMP process on the DLR and the external router may generate *asymmetric traffic* (i.e., when a connection may use an NSX Edge for ingress traffic

and another for egress traffic) this model does not support *stateful services,* which require symmetric traffic for state control. Therefore, in ECMP mode the following stateful services must be disabled: NAT, load balancing, Edge firewall, and all three VPN options.

Some aspects can facilitate this choice:

- Multiple Edges can be deployed in a single NSX deployment (e.g., exclusive Edges per tenant, environment, or line-of-business). A single project can deploy both high availability modes simultaneously at each tenant, environment or line-of-business, as will be shown in Chapter 5, “Industrializing Networks”.

- Stateful services can be deployed through additional NSX Edges on a virtual topology.

Network Address Translation

Although modern IT concepts use resource pooling as an essential characteristic for cloud computing environments, the term has also been applied to *Network Address Translation* (NAT) technologies for some decades. In such scenarios, a set of IP addresses could be used, leased, and reused by a much larger number of hosts.

A conventional networking concept, a network device is performing NAT if it is changing the IP address – source, destination, or both - before it forward the packets through an outgoing interface. Respectively, these operations are called *source NAT* (SNAT) and *destination NAT* (DNAT).

There are multiple motivations for such operations:

- **Internal use of private IP addresses:** Defined on RFC 1918, these private addresses (10.0.0.0 to 10.255.255.255, 172.16.0.0 to 172.16.31.255, 192.168.0.0 to 192.168.255.255) cannot appear in the Internet since they are intended for internal use and, hence, are not routable.

- **IP subnet overlap:** When the same IP subnet address exists in two or more different environments within the same routable domain, such as a data center. Examples include a company merge or an extranet connection between two different corporations.

Note: This case is particular useful in automated environments, as Chapter 5, “Industrializing Networks” will explain.

- **IP Address Sharing (overloading):** NAT can be deployed to make a set of internal users share the same public IP address. In this case, their traffic will be differentiated through a change on the source port of the transport protocol, TCP or UDP, in an operation that is commonly called *Port Address Translation* (PAT).
- **Protocol Fixing:** When an application is using a transport protocol port which is not the standard port used by the application clients with a PAT operation.
- **Application Migration:** When for a certain period of time some application servers will be replaced by another set of servers with another set of IP addresses. In this case, destination NAT is used.

Whatever the reason justifying a NAT configuration in an NSX implementation, the NSX Edge, being the network device dividing what is physical and what is virtual, should be the device responsible for these operations.

Figure 2-15 exemplifies both ways the NSX Edge can implement NAT.

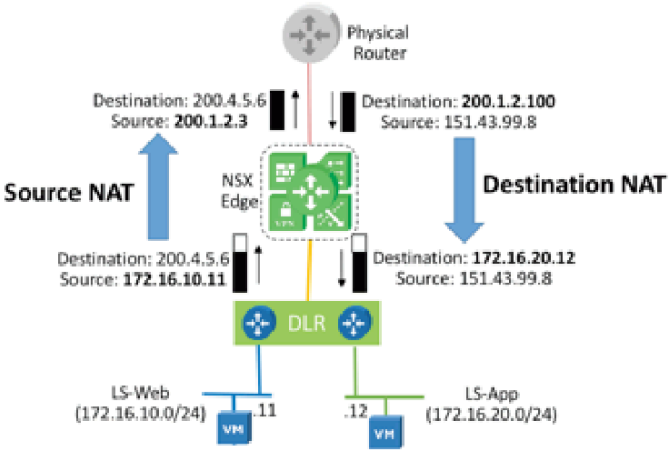


Figure 2-15 VMware NSX Edge Network Address Translation

Note: For the sake of simplicity, the standby NSX Edge is not represented in the figure.

As Figure 2-15 denotes, an NSX Edge can deploy both source NAT and destination NAT through the use of rules. For the packet leaving a VM whose address is 172.16.10.11, the NSX Edge is performing a source NAT operation translating the source address into address 100.4.5.6.

On the other hand, the NSX Edge is also performing a destination NAT operation on a packet directed to original IP address 200.1.2.100. As a result, the packet destination address is transformed to 172.16.20.12, which belongs to a VM connected to logical switch LS-App.

Figure 2-16 depicts the rules that are configured in the NSX Edge to execute such translations.

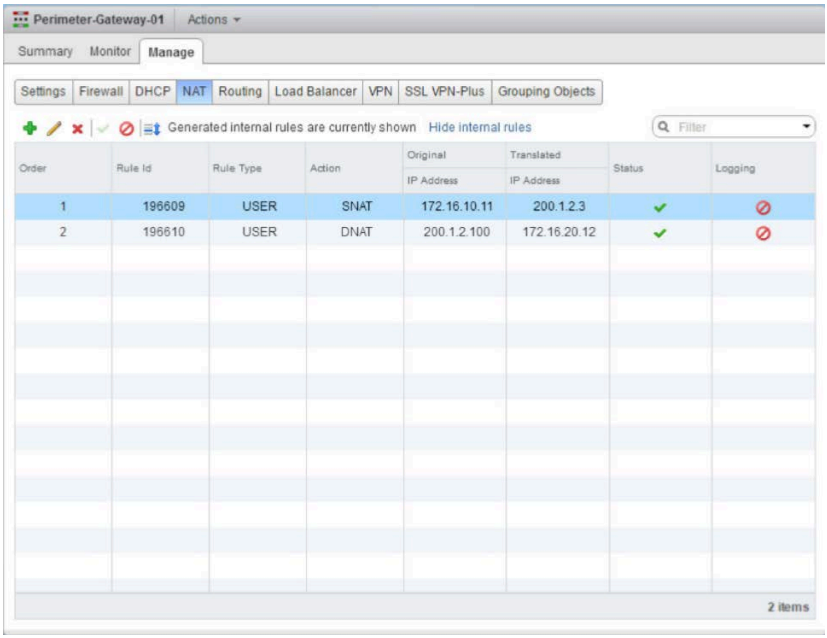


Figure 2-16 NAT Rules Configuration

Note: In an NSX Edge, source NAT is configured on the incoming interface while destination NAT is configured in the outgoing interface.

Server Load Balancing

Server load balancers (SLBs), have been one of the most common network services in data centers since their inception in the middle-1990s. SLBs were created to provide the following services to the exploding e-business market:

- **Application scalability:** Whenever an application server saturates one of its hardware resources (e.g., CPU, memory, storage, or connectivity), all connected application users suffer a performance impact, such as a higher response time. In such scenarios, SLBs allow a group of application servers to share user requests, carefully avoiding this suboptimal state.

- **Application High Availability:** During the e-business bloom, application servers were load balanced through DNS servers. In this case, the same URL (e.g., http://www.acme.com) is registered in a DNS server with multiple IP addresses. As application users request the same URL, they are directed to different IP addresses and their servers. However, if a given server suffers a failure, its IP address is still advertised to a group of users who will then reach nothingness.

- **Content Switching:** In certain occasions, users that access the same application should be treated differently. For example, a user accessing an application through a mobile device should have a different experience from one doing the same on a desktop. Rather than having multiple presentation modules on all servers, an SLB can analyze data above L4 such as browser type or complete URL to determine the best server for each user.

The majority of SLB implementations follow the same basic architecture, which is represented in Figure 2-17.

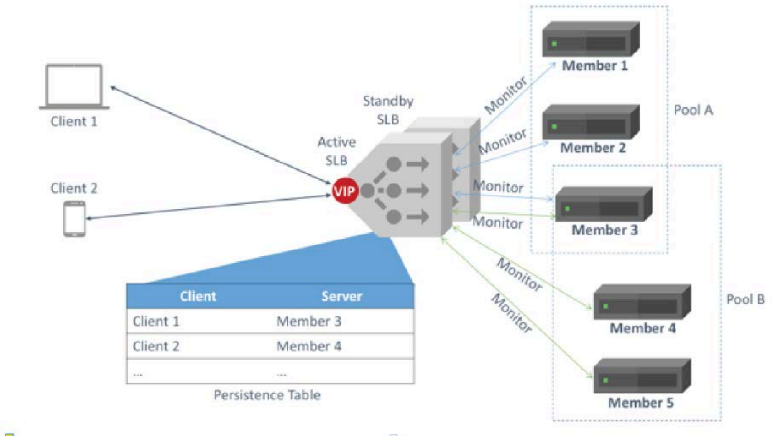


Figure 2-17 Server Load Balancer Architecture

In the figure, the following SLB configuration elements are identified:

- **Member:** Parameter that represents the IP addresses and transport protocol - TCP or UDP - port from servers that will receive the connections dispatched from the SLB.
- **Monitors:** Synthetic requests originated by the SLB to verify whether an application is available on a server. These requests can be as prosaic as Internet Control Message Protocol (ICMP) echo request or as sophisticated as a Hypertext Transfer Protocol (HTTP) GET operation bundled with a database query.
- **Pool:** A group of real servers (i.e., members) that deploy the same application, use the same monitor, and are balanced via an algorithm such as round robin, least connections, or hashing.
- **Virtual IP (VIP):** An IP address that is specifically used to receive application client connections.
- **Virtual Server:** Combines a VIP, a transport protocol, a port or range, and other parameters that identify a particular load-balanced application. It is usually linked to a pool whose members will actually process the dispatched client connections.
- **Persistence table:** An optional SLB component that records client characteristics and their first accessed server. The SLB uses this table to forward posterior client connections to the server previously selected, thus maintaining application consistency. Source IP address, HTTP cookies, and special strings are some of the client characteristics that can be stored on a persistence table.

With the characters properly introduced, the SLB resolution process proceeds as follows:

Step 1. SLB receives a user connection on one of its VIP addresses, analyzes additional parameters to identify the virtual server that will take care of the connection, and verifies whether the user is already registered in the persistence table.

Step 2. If the user is not in the table, the SLB analyzes the pool associated to the virtual server and determines through monitor results which pool members have the application healthily working at that specific moment.

Step 3. Using the virtual server configured load-balancing algorithm, the SLB selects the pool member that will receive the client request.

Step 4. The SLB records the client characteristic and selected member in the persistence table and splices both connections - client-SLB and SLB-member - until the client receives its intended response from the load-balanced application.

After a while, SLBs started to present application optimization features such as *SSL acceleration*. Figure 2-18 portrays the impact of this feature on secure web applications.

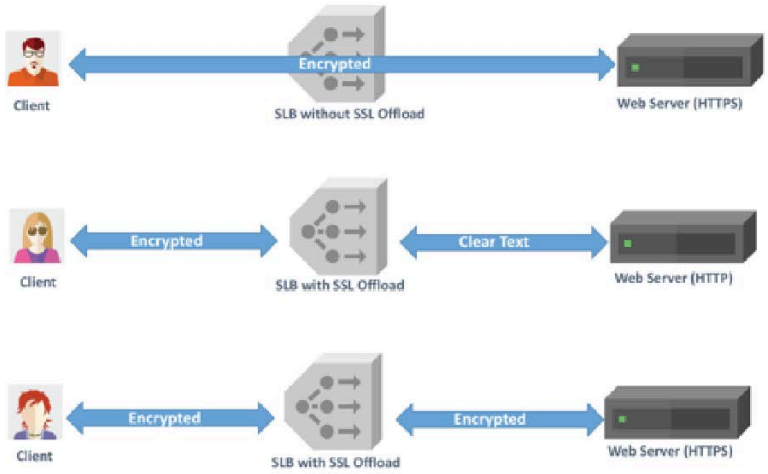


Figure 2-18 SSL Acceleration in Action

The first example shows what happens when an SLB processes a Hypertext Transfer Protocol over SSL (HTTPS) without any SSL acceleration. Because the SLB cannot access any data above L4, which is encrypted by SSL, it cannot perform any layer 5-7 actions such as analysis of URLs or client devices. Additionally, the SLB does not relieve the selected HTTPS server from encryption and decryption tasks, which are often CPU-intensive.

In the second example, the SLB completely offloads the load-balanced web servers from any encryption duties, allowing it to serve more concurrent clients with a better response time. In addition, the SLB device can now process layer 5-7 information to make content-based decisions.

The third example represents a scenario where the SLB creates a new SSL back end session to the web server. This approach is commonly used on applications where encryption is required from client to server because of security or compliance requirements. The SLB can deploy a weaker encryption algorithm to attend such requirements and still save CPU cycles on the load balance servers, reduce the number of public certificates from many to one, and still allow the layer 5-7 analysis of originally SSL-encrypted traffic.

Introducing Logical Load Balancing

In addition to an increasing number of features, SLBs also evolved their form factors through time.

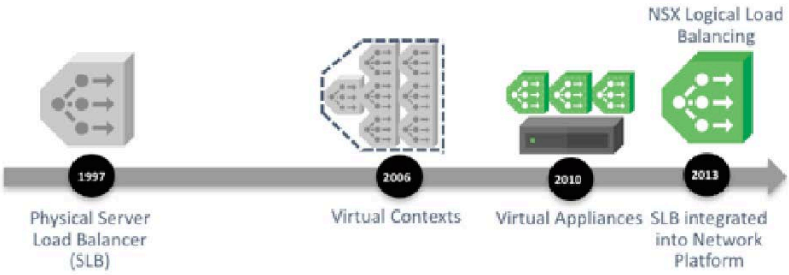


Figure 2-19 The Evolution of SLB Form Factors

As Figure 2-19 exhibits, SLB formats have undergone through a radical evolution over the last two decades.

Achieving independence from solutions based on specific operating systems in 1997, both Cisco and F5 released SLBs based on *physical appliances.* These were devices specifically developed to execute SLB services while connected to a data center network with a predictable performance defined by the appliance hardware characteristics.

As SLB services became extremely popular, some data centers experienced the sprawling of lightly used physical appliances on multiple environments, creating a situation ripe for device virtualization. In 2006, network vendors such as Cisco started to offer *virtual contexts* running on proprietary hardware to optimize the acquisition and deployment of SLBs on large data centers.

Note: During the 1996-2006 period, features such as *web application firewall* (WAF) and *Distributed Denial of Service* (DDoS) defenses were added to SLBs. These devices were renamed *Application Delivery Controllers* (ADCs) to signal this evolution, however this text will continue to use SLB as a generic term for such devices for the sake of simplicity.

With the server virtualization revolution blooming during the first decade of the century, SLBs were one of the first networking services to be virtualized in hypervisor architectures such as VMware vSphere. Initially they were managed as independent devices deployed as virtual machines. This situation changed with the introduction of the *VMware NSX Logical Load Balancer* in 2013, which deployed SLB as an integrated service within a network virtualization platform.

In the NSX architecture, NSX Edges deploy the SLB services through two basic virtual topologies: *inline* or *one-armed*.

Figure 2-20 represents the inline topology.

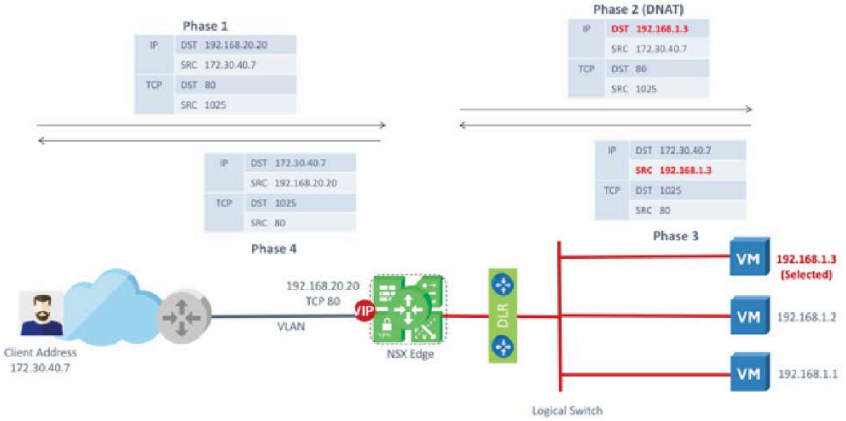


Figure 2-20 VMware NSX Inline Logical Load Balancing

As Figure 2-20 explains, NSX logical load balancing can be activated as another service in a preexistent NSX Edge connecting the virtual network to the physical infrastructure. With the creation of a virtual server using VIP address 192.168.20.20, a client can reach the NSX Edge in what Figure 2-20 calls Phase 1 of the NSX logical load balancing.

Because this is an inline topology, the NSX logical load balancer performs a destination NAT operation, changing the VIP address to the server IP address in Phase 2.

As the selected server responds to the client request, it reverses destination and source addresses in Phase 3. Because the NSX Edge is on the path of the VMs return traffic, it can undo the destination address translation, directing the server traffic correctly to the client IP addresses in Phase 4.

The main advantage from inline topologies is the guarantee that server traffic always returns through the NSX Edge providing SLB services, removing any requirement for network designers to utilize traffic steering techniques. Still, these professionals must remain aware that the same NSX Edge is sharing resources between routing and SLB services to correctly size the NSX Edge so it can fulfill the required performance requisites.

Alternatively, Figure 2-21 portrays how the SLB one-armed topology is applied to VMware NSX.

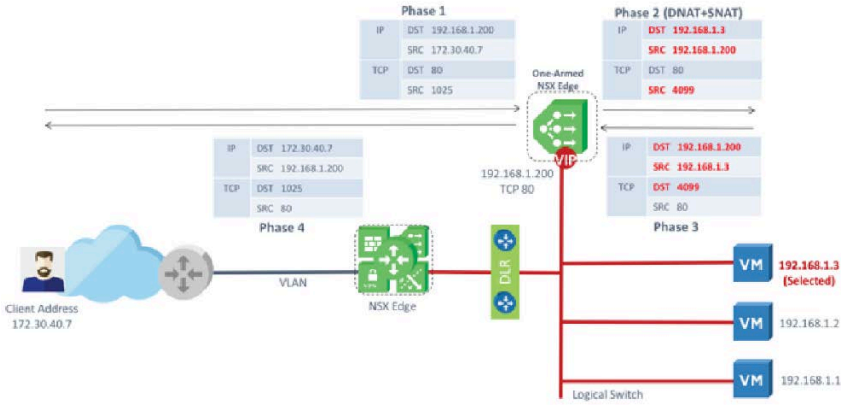


Figure 2-21 VMware NSX One-Armed Logical Load Balancing

In this figure, the SLB service is deployed via a dedicated NSX Edge directly connected to the logical switch shared with its load-balanced servers. As a client reaches the SLB VIP during Phase 1, the virtual device performs *source NAT* in Phase 2. The address translation will assure that the server return traffic will reach the NSX Edge deploying the logical load balancer in Phase 3. All network address translations are undone in Phase 4 to provide a transparent experience to the client.

The main advantage of the one-armed model is its ease of insertion, where an NSX Edge needs just one connection to the same VLAN or logical switch that connect the servers. However, because of the source NAT operation, the servers do not see the client's original IP address, a side effect that may go against some auditing policies.

As an alternative to such scenario, the NSX logical load balancer supports a feature called *X-Forward-For*, which inserts the client original IP address into the namesake HTTP field so it can be properly collected by logging processes running inside the load-balanced servers.

Note: VMware NSX logical load balancing can support servers connected to VLAN or logical switches, as long as they are reachable through IP routing.

Because they can be considered a complete SLB solution for modern data centers, VMware NSX logical load balancers deploy fairly sophisticated features that were not previously discussed such as:

- **Protocols:** TCP, UDP, HTTP, HTTPS, FTP
- **Algorithms:** Round-robin, least connections, source IP hash, URL header
- **Monitors:** TCP, UDP, ICMP, HTTP (GET, OPTION, POST), HTTPS (GET, OPTION, POST)
- **Persistence:** Source IP address, MSRDP, cookie, ssl_session_id
- **Connection Throttling:** Client maximum concurrent connections, client maximum new connections per second, and server maximum concurrent connections
- **Layer 5 to 7 Manipulation:** HTTP and HTTPS request and response headers such as URL block, URL rewrite, and header rewrite
- **Scripting:** Using Application Rules (or simply "AppRules") derived from the well-documented HAproxy software

Note: Multiple configuration example including scripts can be found on VMware Communities at the following URL: https://communities.vmware.com/docs/DOC-31772

In summary, while in terms of features the VMware NSX logical load balancing can work in the large majority of modern data centers, its integration to a virtualization platform responds to fairly recent challenges such as:

- **SLB per application rather than per environment:** VMware NSX can be licensed on a host CPU basis, so there is not a hard limit to the number of NSX logical load balancers that can be deployed on a server virtualization cluster. An administrator may choose to deploy one load balance per application or application layer instead of leveraging the same load balancing appliance for multiple applications.

- **Scale-out model vs scale-up model:** As the number of applications increase, more VMware NSX logical load balancers can be added to each application to avoid expensive and complex upgrades to preexistent appliances.

- **SLB-as-a-Service:** Many applications have seasonal behaviors for which service providers may offer ephemeral application environments. The VMware NSX logical load balancer offers a platform that allows easier deployment and decommission of SLB services when compared to physical solutions.

- **Automation:** Because NSX is centrally managed via NSX Manager, NSX Edges deploying the logical load balancing service can be automatically created and configured via an orchestrator or cloud management stack such as VMware vRealize® Automation™ or OpenStack.

- **Operation Intelligence into Application:** The NSX application programming interface (API) allows software developers to include changes to logical load balancing features, such as resizing and new instance inclusion to a pool, as part of application software code.

Other Interesting Edge Network Features

Before discussing NSX security services in Chapter 3, it is helpful to also explain two commonly overlooked NSX Edge services that can be extremely useful for virtual network topologies.

The first one is *Dynamic Host Configuration Protocol (DHCP)***,** which allows the NSX Edge to provide the following configurations for DHCP clients that are directly connected to an NSX Edge internal interface:

- **IP address:** From a predefined IP address pool or through a static IP-MAC mapping.
- **Subnet mask:** Must be the same configured on the NSX Edge internal interface.
- **Default gateway:** Derived from the NSX Edge internal interface IP address.
- **Primary and secondary name servers:** For hostname to IP address resolution
- **Domain name:** Optional parameter for the DNS server
- **Lease time:** Duration of DHCP-provided configuration

VMware NSX DHCP allows virtual or physical workload IP addressing to be automatic, removing the need for multiple interactions between an orchestrator and the server virtualization infrastructure. It is a well-known protocol, bringing operational familiarity to these scenarios. Additionally, the NSX Edge Services Gateways can deploy *DHCP relay,* which directs local address requests to a centralized DHCP server.

Another service available from the NSX Edge is *Domain Name System* (DNS) relay. When configured, the NSX Edge can receive name-to-IP-address resolution requests from VMs and relay them to external DNS servers. Upon receiving the responses, the NSX Edge caches their mapping to promptly respond to other requests in the future.

What Did This Chapter Achieve?

Figure 2-22 represents in a single virtual network topology the main concepts discussed in this chapter.

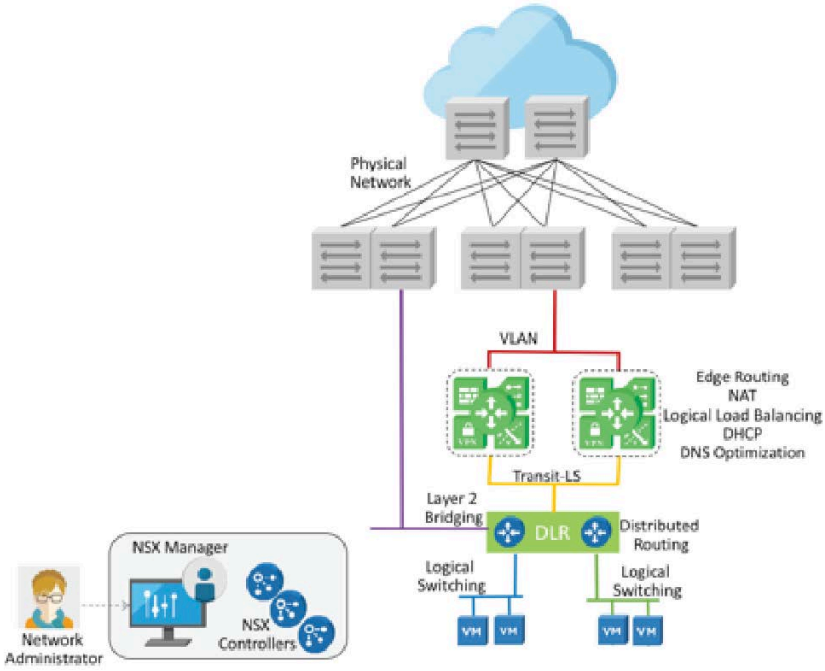


Figure 2-22 VMware NSX Virtual Networking Elements

In summary, they are:

- **Centralized Provisioning:** All virtual network elements are configured through the NSX Manager, which transparently informs the NSX Controller cluster about the operations it should deploy on each host to create the virtual topology.
- **Logical switching:** Allows L2 connectivity between virtual machines through an overlay protocol such as VXLAN, only requiring IP unicast routing from the physical network.
- **Distributed logical routing:** Deploys IP routing inside each of the hypervisors that are part of a VMware NSX transport zone.
- **Edge services:** Enables routing to the external world, network address translation, load balancing, DHCP server, and DNS optimization.

Virtualization as a Security Control

> *"A policeman sees a drunk man searching for something under a streetlight and asks what the drunk has lost. He says he lost his keys and they both look under the streetlight together. After a few minutes the policeman asks if he is sure he lost them here, and the drunk replies, no, and that he lost them in the park. The policeman asks why he is searching here, and the drunk replies, 'this is where the light is'"*
> *unknown author*

It is a massive understatement to declare that security is a broad topic. It deals with human creativity, which unfortunately knows no limit for twisting, exploiting, and destroying.

Information security no different. Although the Internet produced invaluable benefits such as free communication and collective knowledge, people and corporations face daily risks in the form of viruses, worms, and many other forms of attacks from billions of interconnected devices. Connectivity and security may walk hand in hand but *rarely* in the same direction.

It is disheartening to witness the sprawl of major data breaches and sophisticated hacker attacks that are serious enough to end careers and take companies out of business. While there is not a perfect solution for all of these assaults, their frequency is giving a sign to the security industry that it may be suffering from the above-described observational bias called “streetlight effect”.

VMware NSX was conceptualized and developed during a period of deep introspection and rethinking of IT. This network and security virtualization platform embodies radical new principles such as *Zero Trust* in order to bring simplicity and dynamism to security designs.

Like many other advanced virtualization technologies, some NSX functionalities may not be directly mapped to physical counterparts. Before positioning features such as distributed firewall, security partner integration, identity firewall, service composer, Edge firewall, and virtual private networks (VPNs), this chapter explores the challenges that long-established security paradigms are facing in modern IT environments.

With this approach, the chapter intends to explain why NSX is not only a solution that can uniquely and effectively address security in virtualized environments, but also a flexible platform that can actually foster a more effective kind of security architecture.

The Evolution of Firewalls

Data centers are expected to be much more secure than any other IT environment. Most corporations have consolidated their computing and data storage resources in data centers with the intent of protecting these assets from more trivial attacks on remote premises.

Among the myriad information security systems found in a first-class data center, there is certainly one that reigns supreme in the minds of IT architects: the all-powerful *firewall*. Named after a wall built to inhibit or prevent the spread of fire within a building, a firewall can be defined as a network device whose main objective is to control traffic between *network segments* in the context of information security. Based on administrator-defined rules, a firewall analyzes data packets and decides whether or not they should be transported from one network segment to another.

The main objective of network segmentation is to achieve logical separation of applications with different sensitivity levels or that belong to different owners, or tenants, sharing the same network infrastructure.

Statefulness Defined

The first generation of firewalls simply implemented *packet filters,* making their forwarding decisions one packet at a time. Because these devices were not able to recognize some simple attacks such as the one portrayed in Figure 3-1, they are also known as *stateless firewalls.*

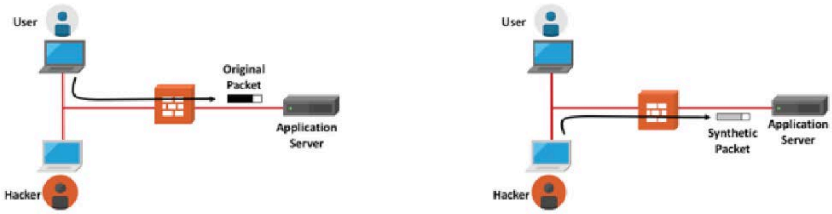


Figure 3-1 Stateless Firewall in Action

In the left side of Figure 3-1, the stateless firewall applies packet filtering against the following fields in the IP packet: destination and source IP addresses, destination and source transport protocol ports, and transport protocol. Per definition, the actions taken by a stateless firewall are solely dependent on such parameters.

As shown in the right side of the figure, a hacker could generate a synthetic sequence of packets reflecting valid parameters, making it impossible for the stateless firewall to discriminate them from genuine user traffic and exposing the application server to malicious activity.

Note: There are multiple ways a hacker can discover valid connection parameters to fool a stateless firewall, including traffic sniffing - which was very easy in early Ethernet networks - or false ARP messages advertising the hacker machine MAC address as the one associated with the user default gateway. In the latter method, the hacker machine would still send legitimate user traffic toward the destination IP address, but would also be able to monitor all packets from this communication. This is known as man-in-the-middle attack.

A firm step into the evolution of network security was taken with the creation of *stateful firewalls.* And as Figure 3-2 illustrates, such a device takes other conditions in consideration before making a forwarding decision.

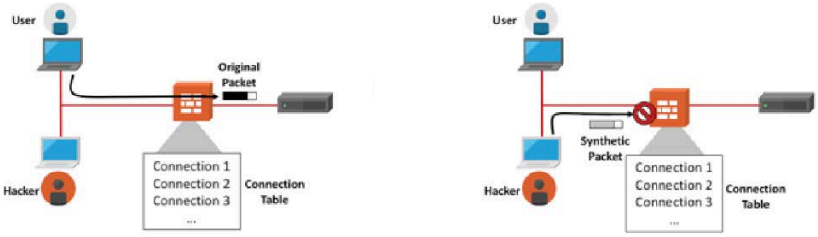


Figure 3-2 Stateful Firewall in Action

On the left side of Figure 3-2, it is shown that the stateful firewall is keeping a *connection table.* The connection table allows the recognition of packets that belong to an established L4 connection between hosts located in different network segments. As represented on the right side of the figure, when a hacker tries to send a synthetic packet for misuse of a protected segment resource, the firewall checks it against its connection table to determine whether the packet is the start of a new connection, part of an existing connection, or not part of any connection.

Because the hacker is trying to emulate IP packets that belong to a connection, a stateful firewall deploys a dynamic tracking mechanism called *Stateful Packet Inspection* (SPI) to identify bona fide packets from the user connection and block malicious traffic. To accomplish this objective, SPI holds in memory significant attributes of each connection including IP addresses, transport protocol ports, and sequence numbers in the case of TCP connections.

Application Firewalls and Next-Generation Firewalls

Further along in the evolution of network security, more modern devices called *application firewalls* gained visibility in connection parameters above L4 using *Deep Packet Inspection* (DPI). This mechanism is capable of recognizing upper-layer metadata from popular applications and protocols.

A significant benefit of an application firewall is its ability to validate whether a higher-layer protocol (e.g., HTTP, FTP or DNS) is actually used within an L4 flow with its assigned destination transport protocol port. With such capability, these security devices can reject attacks that may be hiding behind common L4 connections.

Finally, devices called *next-generation firewalls* incorporated other functions beyond DPI such as:

- **Intrusion Detection System (IDS):** Analyzes packet data payload to detect malicious activity or policy violations based on regularly-updated attack signatures. An IDS reports the attack to a logging server and usually does not react to the attack.

- **Intrusion Prevention System (IPS):** Similar to an IDS, but it can block a connection as soon as an attack is identified among the list of attack signatures.

- **User Identity Management:** Provides the binding of user or group identifiers to IP or MAC addresses, allowing simpler and more flexible firewall rules.

- **Web Application Firewall (WAF):** According to the Open Web Application Security Project (OWASP), a WAF is a specialized firewall for HTTP applications. Because it closely monitors, analyzes, and blocks traffic based on specific HTTP filters, it can prevent attacks such as SQL injection, cross-site scripting (XSS), file inclusion, among others.

- **Antivirus (AV):** Used to prevent, identify, and remove malicious software such as viruses and worms that may be contained in computer files.

- **URL Filtering:** Provides granular control over web browsing activities via parsing and identification of HTTP Uniform Resource Location (URL) strings.

  In many next-generation firewall implementations, the potential exists for activation of these advanced features to impact the appliance performance in terms of throughput, concurrent connections, and new sessions per second.

Demilitarized Zone

In a military context, a *demilitarized zone* (DMZ) is a neutral area between two or more countries where no combat action is allowed according to a multilateral treaty. In information security, a DMZ is defined as a network segment whose main function is to expose services to external networks - such as the Internet - without compromising other more secure segments.

Figure 3-3 depicts a firewall delimiting a DMZ.

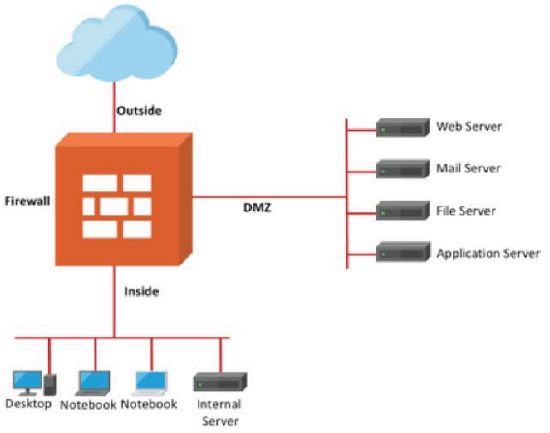


Figure 3-3 A Demilitarized Zone

In Figure 3-3, the DMZ is defined by a mid-security level network segment connected to a third firewall interface. This allows a single firewall to enforce security rules to data traffic exchanged among all three interfaces: Outside-DMZ, DMZ-inside, and outside-inside.

Additional DMZs can be created as more interfaces are connected to distinct network segments in the topology shown in Figure 3-3.

Classic Data Center Security Designs

Any design is a direct result of intent, available tools, best practices, and existing limitations. For such reasons, no one should criticize a traditional design without understanding the problem it sought to solve and the context behind its main decisions.

The same mindset should be adopted in the analysis of the classic data center security designs in the last two decades. Figure 3-4 starts the journey showing a traditional design from the 1990s.

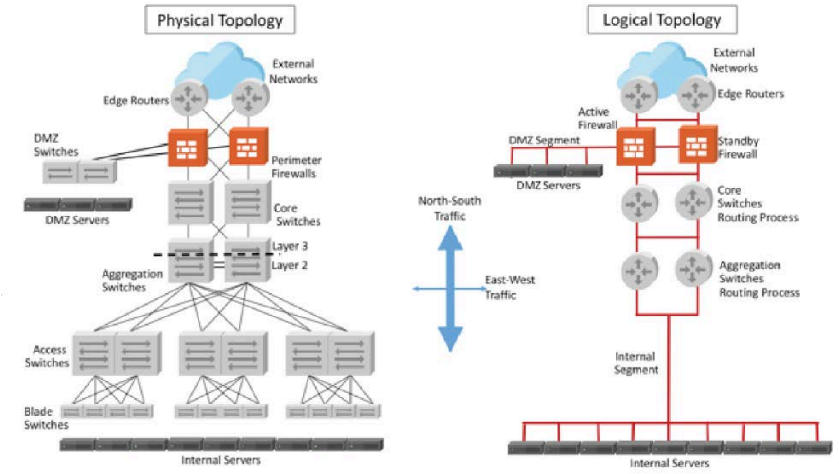


Figure 3-4 Perimeter-focused Data Center Security Design

The left side of the figure represents a typical physical network topology from this period. When e-business was growing, most applications were still *monolithic* (i.e., single-tiered) and north-south traffic (i.e., client-server) was prevalent over east-west traffic (i.e., server-server) in data centers. *Firewalls* were deployed at the data center perimeter in order to keep threats as far as possible from the precious internal servers.

The corresponding logical topology is shown on the right side of the figure, highlighting a single internal network segment. If new VLANs were added to the network infrastructure, the firewalls would not be capable of controlling east-west traffic due to their position in the network. Consequently, from a firewall perspective, a VLAN did not create a new network segment in this design.

In parallel, servers with externally accessible content were put on a DMZ segment, which in most cases required separate physical switches. For this requirement, many DMZ implementations created a resource silo inside the data center which could not share hardware resources - such as physical switches and even servers - with other network segments. This caused management complications and excessive infrastructure costs for the sake of isolation.

Note: As assumed in the logical topology, the firewalls are deploying active-standby high availability while the L3 switches, both core and aggregation, are relying on ECMP redundancy. Both concepts were explained in Chapter 2, “Rethinking Network Constructs”.

The 2000s saw two significant shifts in terms of computing: the popularization of *multitier applications* and the increasing adoption of *server virtualization*. As server communication changed, many data centers adopted the security design presented in Figure 3-5.

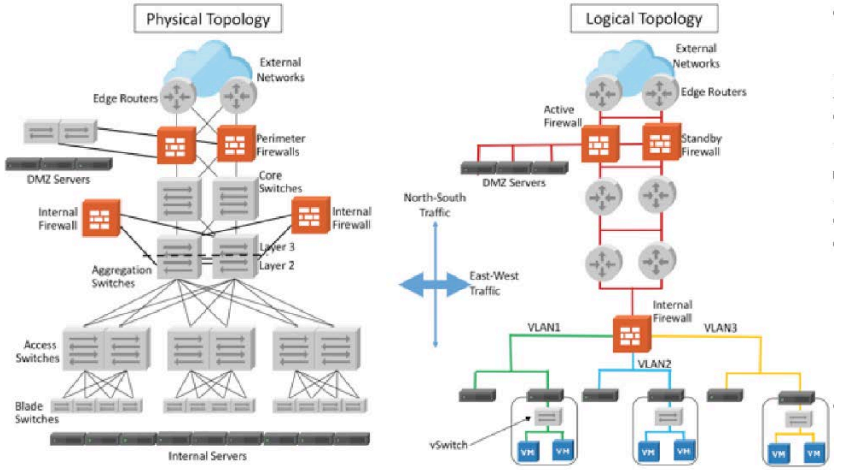


Figure 3-5 Dual-Layer Data Center Security Design

In the physical topology on left side, an internal firewall was added to enforce security rules between distinct internal network segments. Due to the evolution of L2 security features in switches, most security teams started to consider VLANs as an acceptable traffic isolation technique. The VLAN eventually became synonymous with network segments for security policies; however, some data centers preferred to maintain their DMZs in separate physical switches to follow their security policy directives.

Note: Depending on the data center sizing, some designs utilize a single firewall layer, changing its physical position toward the aggregation switches providing access to all VLANs.

On the right side of Figure 3-5, the logical topology uncovers the function of the internal firewall. Because application architectures were changing from monolithic to multitier, server-to-server traffic was increasing substantially. In this scenario, the internal firewall was responsible for handling east-west traffic between different network segments (i.e., VLANs) while the perimeter firewall kept its focus on north-south traffic.

From a functional perspective, next-generation features such as IPS and WAF were activated on the perimeter firewalls to increase protection

against Internet-borne attacks. On the other hand, because they were handling relatively more secure network segments, the internal firewalls deployed simpler stateful security rules to preserve performance for the ever-increasing traffic between network segments.

Figure 3-5 also depicts a simplified view of server virtualization networking, with virtual switches controlling virtual machine traffic within hypervisors. Because vSwitches then only worked as L2 networking devices, VMs were put on distinct VLANs whenever their traffic needed the inspection of a firewall.

Anatomy of a Modern Cyber Attack

Cyber attacks have significantly evolved since computing left mainframe rooms to inhabit offices and households around the world. Through technical knowledge, collaboration, and social engineering, hackers have been further expanding their attacks in number and impact.

The following chronological list illustrates the evolution of cyber attacks that are collectively known as *malware*:

- **Viruses (1982):** Malicious programs that are designed to affect computer operation and performance**.** They require user activation and are usually shared through an executable file.
- **Worms (1988):** Similar to viruses, worms are computer programs that can also subdue computer operation and performance. Worms can replicate over a computer network without any user interaction. As a consequence, they can cause major disruptions on multiple systems simultaneously.
- **Ransomware (1989):** Software whose intention is to encrypt private data or threaten to expose it publicly if a ransom is not paid.
- **Trojans (1998):** Computer programs that disguise themselves as useful applications, but are developed to cause computer malfunctions or undetected data theft. Their name come from the Greek tale about the Trojan War.
- **Spyware (2000):** Software that does not cause any harm to a computer system, but instead gathers information about a person or organization without their knowledge or that asserts control over a device without its user's knowledge.
- **Rootkits (2005):** Set of malicious programs that can facilitate administrative (i.e., root) access to unauthorized components of a computer operating system.

Modern cyber attacks, also known as *advanced persistent threats* (APTs), use a combination of malware code and undetected hacking activity via an external *command and control system* for long periods of time. Their main objective is to provoke data breaches on private companies, governmental organizations, and influential people.

APTs follow a sequence that is commonly referred as *cyber kill chain* – a term inspired in the structure of a military attack and first defined by global security and aerospace company Lockheed Martin.

> **Phase 1. Infiltration:** Malware is delivered inside of an organization. Its landing point is usually not the final target of the attack, but the weakest part of the organization.
>
> **Phase 2. Propagation:** Malware escalates privileges inside computer systems, installs access for the command and control infrastructure, and leverages lateral movement within the organization network.
>
> **Phase 3. Extraction:** Attack breaks into data stores, initiates network eavesdropping, and extracts critical data.
>
> **Phase 4. Exfiltration:** APT bundles data and obfuscates its origins. It sends the data to an external site in encrypted form and erases any evidence from the operation.

According to Tom Corn, Senior VP of Security Products at VMware, more than eighty percent of all security investment is spent on Phase 1 (i.e., infiltration) while only twenty percent in the other three phases of the cyber kill chain. He continues:

> *"It's extraordinarily difficult to put controls inside the environment, to compartmentalize critical areas of risk, and to have the visibility and controls in the right places to address these other stages. But without addressing them, we are doomed to continue on our current trend."* [1]

The re-architecture of security controls suggested would break the kill chain of APTs as well as other attacks such as *WannaCry*, a sophisticated combination of ransomware and worm that moved almost freely inside local networks and encrypted data in more than 300,000 computers in 2017.

In other words: *Hasn't phase 1 of all attack kill chains been the "streetlight" for most traditional security designs?*

[1] https://www.vmware.com/radius/virtualization-and-ending-security-theater/

During phase 2 of the APT kill chain (i.e., propagation), it is clear that all workloads connected to a single VLAN will be vulnerable to the attack lateral movement. Because traffic in these L2 structures is allowed by default, positioning stateful firewalls to handle inter-VLAN communication only solves a small part of the propagation problem, as Figure 3-5 suggests.

Although some valiant efforts such as *Private VLANs* (PVLANs) and *VLAN Access Control Lists* (VACLs) were created to provide L2 traffic control, they all shared common drawbacks: implementation complexity along with the requirement that every L2 device – both physical and virtual - must deploy them to avoid data leaks.

Undoubtedly, the time was right for a new security paradigm.

The Zero Trust Model

Introduced by analyst firm Forrester Research in 2011, *Zero Trust* is a security architecture that preaches that no resource, including internal ones, should be given complete trust. With such "never trust, always verify" ethic, Zero Trust is especially designed to address all phases from attack kill chains, including propagation through lateral movement.

Zero Trust intends to optimize security designs "from the inside out" in order to avoid network topologies that enable exchange of toxic data. Consequently, new workloads should automatically follow security policies that only allow expected behavior: everything else must be blocked.

The next section will present an example of the Zero Trust model applied to daily life to clarify how it can be effectively employed in a data center security design.

A Real World Analogy: Secure Airport

Because of terrorist acts that occurred during the last decades, a well-known environment was obligated to adopt Zero Trust principles in its security architecture: the airport.

Modern airports deploy a security design to safely guide a traveler from an insecure zone - where physical access is available to virtually anyone in the planet - to their correct airplane.

Figure 3-6 presents a simplified model of such design.

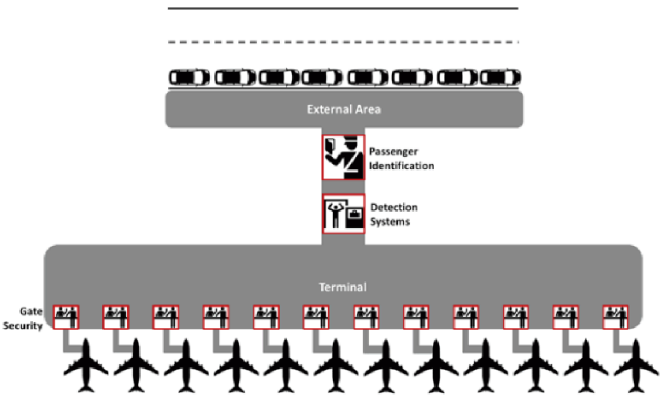


Figure 3-6 Airport Security Design

In this figure, the following security systems are highlighted:

- **Passenger Identification:** In this process-based system, an airport official verifies the passenger's flight information (e.g., flight number, terminal, airport, and boarding time) and identity (i.e., documentation). A passenger will only be allowed to go the next security system if all presented information complies with the airport security policies.

- **Detection Systems:** This security stage leverages different technologies (e.g., metal detectors and backscatter X-ray) to detect possible weapons that the passenger may be carrying.

After this point, the airport authorities still do not consider travelers in the terminal fully trustworthy for one reason: human creativity. For example, a possible hacking activity could be a passenger using valid credentials to board a domestic flight but actually trying to board an international one.

A theoretical solution to avoid such problems could be the creation of terminals with different security levels (e.g., international vs. domestic terminals). However, this *segmentation strategy* would not block a traveler arriving at the international terminal from trying an unauthorized connection via a departing international flight.

To efficiently address these and all other possible unknown unsecure situations, airport security designers decided to insert a second-level of identity and destination checking as close as possible to the airplane. Even though the *gate security* system is executing the passenger identification again, its position helps thoroughly enforce the airport security policy.

It is fairly easy to correlate this airport security design with the one of a data center. As a healthy exercise, try to mentally make this comparison before reading Table 3-1.

Table 3-1 Airport vs Data Center Comparison

Airport	Data Center
Passenger	Application traffic
Airplanes	Servers
External area	External networks (e.g. Internet)
Passenger identification	Perimeter firewall
Detection systems	IPS, IDS, and next-generation firewall
Terminal	Internal network segment
Passenger boarding	North-south connection
Passenger connection	East-west connection
Gate security	*VMware NSX distributed firewall*

The next section provides more information about the NSX mechanism that can provide the same strict control in a data center an airport offers: *the distributed firewall.*

Introducing the VMware NSX Distributed Firewall

One of the most celebrated features of VMware NSX, the *distributed firewall* (DFW) is in essence a hypervisor kernel-embedded stateful firewall that provides visibility and control for virtualized workloads.

Figure 3-7 illustrates how the distributed firewall fits into a data center security design.

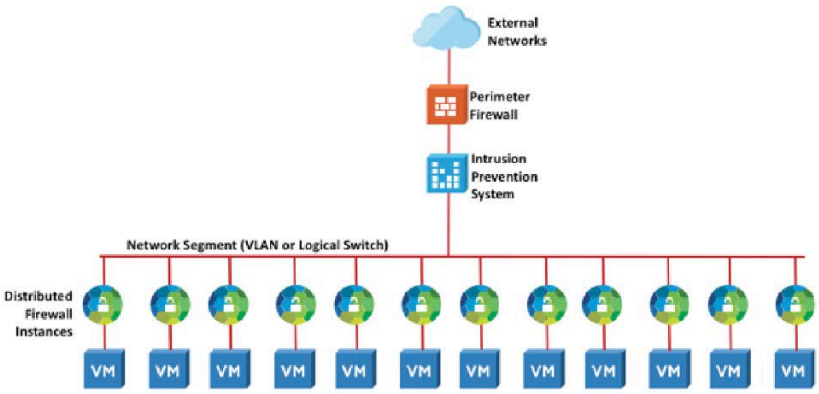


Figure 3-7 Data Center Security Design with VMware NSX Distributed Firewall

In this figure, the distributed firewall stands as close as possible to the virtual machines. Distributed firewall rules are enforced at the *virtual network interface controller* (vNIC) level to provide consistent access control even if the virtual machine is migrated via vMotion to another host and regardless of the network segment it is connected to (e.g., VLAN or logical switch).

There is an enormous advantage to having a firewall residing outside the virtual machine guest operating system. The hypervisor controls the I/O path to and from the vNIC and uses it to deliver security services - like the distributed firewall - which is out of the attack surface of operating system malware such as rootkits. Because the NSX DFW is stateful, it also monitors the state of active connections and uses this information to determine which network packets traverse the virtual machine vNIC.

Note: In the airport analogy, the gate security system must also be stateful, or else someone with a copy of a valid ticket would be allowed to board the plane after the proper passenger has already entered the aircraft.

In NSX, the configuration of distributed firewall rules is organized in *sections* as Figure 3-8 exhibits.

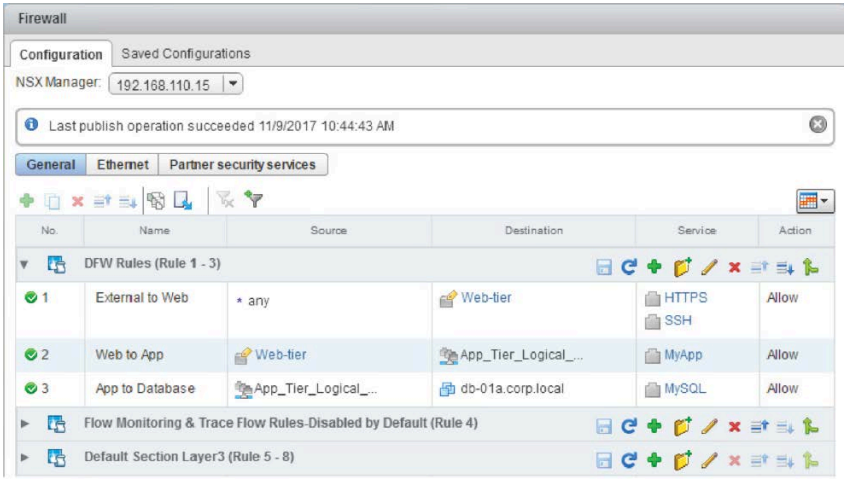


Figure 3-8 VMware NSX Distributed Firewall Configuration

After a quick glance at Figure 3-8, it is possible to perceive that the DFW rules are not restricted to IP addresses and transport protocol ports. As the figure shows, section **DFW Rules** has three rules:

1. **External to Web:** Allows any source to access security group **Web-tier** - which includes all VMs with the string "web" on their names - through HTTPS (i.e., TCP port 443) and SSH (i.e., TCP port 22)

2. **Web to App:** Allows the same security group **Web-tier** to access any VM connected to logical switch **App_Tier_Logical_Switch** through a customized service called **MyApp**.

3. **App to Database:** Allows any VM connected to logical switch **App_Tier_Logical_Switch** to access virtual machine **db-01a.corp.local**

Note: Security groups will be addressed in more detail in a section "Security On-Demand: Service Composer".

Now it is time to take a quick look under the hood of this flexible firewall. A distributed firewall instance, running on a virtual machine vNIC, contains two tables: a *rule table* that stores all configured rules - such as the ones listed on Figure 3-8 - and a *connection tracker table* that caches flow entries for rules with **Allow** action.

NSX distributed firewall rules are enforced in ascending order according to their number parameter. Each packet traversing a distributed firewall instance is checked against the top rule before moving down the subsequent rules. The first rule that matches the packet parameters will be enforced. No other subsequent rule will have influence over that packet.

The last distributed firewall rule is the default policy rule, which enforces a defined action to packets that do not match any rule above it. By default, this rule is configured with an **Allow** action, but it is possible to change it to a **Block** action.

For the sake of completeness, Table 3-2 enlists all parameters from a distributed firewall rule.

Table 3-2 VMware NSX Distributed Firewall Rule Parameters

Rule Parameter	Description
Number	Determines the order of execution of the rule
Name	User-defined string that identifies the rule
Rule ID	Unique system-generated identifier for the rule
Source	Specify the traffic source described in the rule, which may include: cluster, data center, distributed port group, IP set, legacy port group, logical switch, resource pool, security group, vApp, virtual machine, vNIC, IP address (IPv4 or IPv6)
Destination	Specify the traffic destination described in the rule, which may include: cluster, data center, distributed port group, IP set, legacy port group, logical switch, resource pool, security group, vApp, virtual machine, vNIC, IP address (IPv4 or IPv6)
Service	Specifies a port protocol or a combination of ports on predefined list. It can also be customizable
Action	Defines the rule behavior when the described traffic is identified. It can be: Allow, Block, or Reject (sends RST message for TCP connections or ICMP messages for UDP, ICMP, and other IP communications
Applied To	Narrows down the scope of the rule. It can be: all clusters, all NSX Edges, cluster, data center, distributed port group, NSX Edge, legacy port group, logical switch, security group, virtual machine, vNIC
Log	Allows that traffic that matches the rule is logged
Direction	In/Out (bidirectional), In (ingress), Out (egress) from the virtual machine perspective
Packet Type	IP version: any, IPv4, or IPv6
Tag	String used for automation over rules (NSX API)
Comments	Enables the insertion of administrator comments for better reading of rules

Note: As with most software-based solutions, the neck-breaking evolution of NSX can also be verified in the release of its *context-aware Firewall*. It can be understood as an enhancement of DFW to increase visibility at the application level (layer 7) to monitor workloads with a more granular perspective. In essence, context-aware firewall identifies applications through a distributed Deep Packet Inspection (DPI) engine and enforces rules that are based on more information than 5-tuples. Some of the applications recognized are Microsoft Active Directory, Advanced Message Queueing Protocol (AMQP), Common Internet File System (CIFS), Citrix Independent Computing Architecture (ICA), DNS, File Transfer Protocol (FTP), HyperText Transfer Protocol (HTTP), Internet Message Access Protocol (IMAP), KERBEROS, Lightweight Directory Access Protocol (LDAP), PC over IP, Remote Desktop Protocol (RDP), Session Initiation Protocol (SIP), SQLnet, Secure SHell (SSH), and Secure Sockets Layer (SSL). Please refer to the VMware documentation to verify whether content-aware firewall is available in the NSX version in use.

VMware NSX Security Partners Integration

VMware NSX is considered a complete security platform because of its extensibility capabilities, which allow seamless integration with leading vendors of the security industry. In all of those integrations, NSX allows for security policies to be automatically enforced on VMs while avoiding complex traffic steering via IP routing or VLAN manipulation.

There are two methods of third-party integration in VMware NSX: network introspection and guest introspection.

Network introspection leverages Network-Service-Extensibility (NetX), which is a vSphere framework that enables automatic traffic steering inside of an ESXi host. An NSX administrator takes the following steps to deploy network introspection with security partners:

Step 1. The third-party management console registers its services with NSX Manager.

Step 2. NSX Manager provisions a *third-party service VM* on each host to deploy the security partner specialized services on the selected virtualization clusters.

Step 3. The service VMs register themselves with the third-party management platform to obtain licenses and corresponding security policies.

Step 4. NSX Manager deploys security policies distributed firewall and traffic steering rules that will dynamically be applied to virtual machines running on the previously selected clusters.

The NetX agents that provide traffic steering to the service VMs are called *DVfilters*. They reside in the stream between a vNIC and its connected virtual switch and behave as described in Figure 3-9.

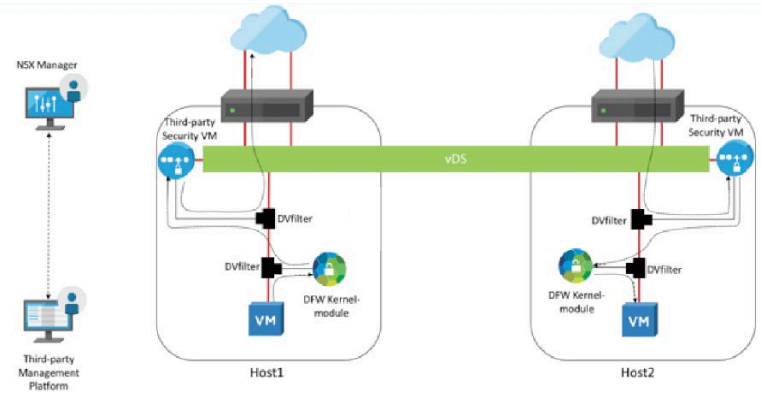


Figure 3-9 Network Introspection in Action

In Figure 3-9, Host 1 has a VM that is sending a packet toward a vDS. The first DVfilter redirects traffic to the distributed firewall kernel process while a second DVfilter steers the packet to the service VM that is performing the third-party security servers. On the other hand, Host 2 is enforcing the inverse order of traffic steering toward the VM that is running inside of it.

Note: Traffic redirection to service VMs does not leverage a standard connection to the virtual switch; it occurs through an internal construct using hypervisor-based memory sharing space called *Virtual Machine Communication Interface* (VMCI).

Besides using traffic steering via network introspection, NSX also provides security partner integration through a method called *guest introspection.* Guest introspection requires the installation of a thin agent in each VM that will use the third-party service. Installing guest introspection on a cluster also deploys a new vSphere Installation Bundle (VIB) and a third-party service VM on each host.

Guest introspection offloads operating system monitoring and file scanning tasks to the dedicated service VM. It is mainly used on integrations with antivirus solutions and, due to its architecture, is also called *agentless antivirus*.

Unlike an antivirus agent running on a VM that can be turned off at any time, a guest introspection service VM can continuously update virus signatures and provide the most current defenses to virtual machines as soon as they come online. In addition, the service VM is not prone to attacks, does not require manual deployments, and considerably reduces resource consumption on the guest VMs.

Note: Guest introspection supports the Windows introspection drivers that are included in VMware Tools but are not part of the default installation. It also supports Linux VMs for antivirus integration only.

Table 3-3 lists the main security partners that offer integration with NSX. For the latest partner integration list please visit - https://www.vmware.com/products/nsx/technology-partners.html

Table 3-3 VMware NSX Security Partners

Security Partner	Network Introspection Services	Guest Introspection
CheckPoint	Next-Generation Firewall	
Fortinet	Next-Generation Firewall	
Intel Security	IPS and IDS	Agentless Antivirus
Palo Alto Networks	Next-Generation Firewall	
Rapid7		Vulnerability Management
Symantec	IPS and IDS	Agentless Antivirus
Trend Micro	IPS and IDS	Agentless Antivirus

Note: Always refer to VMware and security partner documentation to verify the features availabilities on the versions in use.

One of the most interesting aspects of the integration of NSX with third-party solutions is that NSX can provide an automatic response if the third-party solution finds out that the guest VM was compromised (e.g., VM infected with a virus). The third-party management console can attach a *security tag* to the VM, signaling to NSX that it should be isolated from other VMs that are connected the same logical switch or VLAN.

Going back to the airport analogy, think of these security partner integrations as an advanced gate security system (e.g., x-ray system or metal detector) for special flights depending on destination or presence of passengers of interest.

Security On-Demand: Service Composer

The NSX Service Composer allows the creation of extremely flexible security policies. This service enables administrators to define *security policies*, including distributed firewall rules and third-party integration to virtual machines organized in *security groups*.

To create a security policy, the Service Composer wizard follows five steps:

1. **Name and Description:** Besides assigning the initial name and description to the security policy, this step also allows the policy to inherit characteristics from other policies. It also allows the assignment of a weight number to define precedence between security policies applied to the same virtual machine.

2. **Guest Introspection Services:** Integrates third-party services such as antivirus and vulnerability management to a group of VMs.

3. **Firewall Rules:** This step creates distributed firewall rules with a security group serving as source or destination.

4. **Network Introspection Services:** Integrates third-party services such as next-generation firewall and IPS to a group of VMs.

5. **Ready to Complete:** Summarizes the security policy proposed configuration and creates it.

After a security policy is created, an administrator can apply it to one or more security groups. A security group can be formally defined as a flexible container for VMs that can be populated through:

- **Dynamic membership:** Match-any or match-all criteria which includes guest operating system name, computer name, VM name, security tag, and entity membership. Membership includes other security groups, clusters, logical switches, legacy port groups, vApps, data centers, IP sets, directory groups, MAC sets, security tags, vNICs, virtual machines, resource pools, or distributed port groups.

- **Objects inclusion list:** Statically *includes* in the security groups entities such as other security groups, clusters, logical switches, legacy port groups, vApps, data centers, IP sets, directory groups, MAC sets, security tags, vNICs, virtual machines, resource pools, or distributed port groups.

- **Objects exclusion list:** Statically *excludes* from the security groups entities such as other security groups, clusters, logical switches, legacy port groups, vApps, data centers, IP sets, directory groups, MAC sets, security tags, vNICs, virtual machines, resource pools, or distributed port groups.

Figure 3-10 explains the relationship between security policies and security groups.

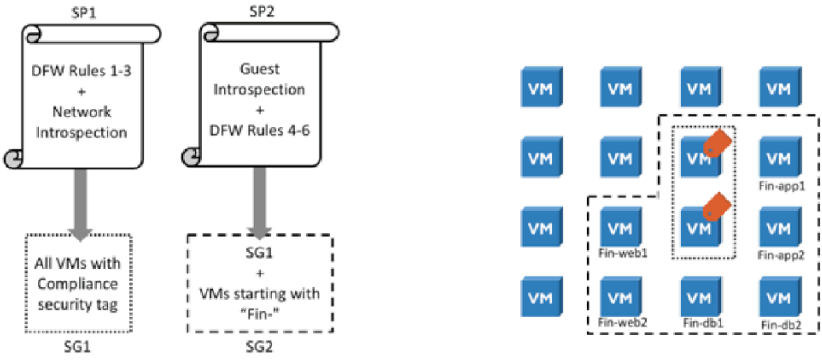


Figure 3-10 Security Policies and Security Groups

In Figure 3-10, two security groups were created via Service Composer: **SG1** - which groups all VMs with a "Compliance" tag - and **SG2** - which encompasses all VMs on SG1 plus all VMs whose name begin with "*Fin-*".

Two security policies are then respectively applied to SG1 and SG2: **SP1,** which applies distributed firewall rules as well as a next-generation firewall rules to all VMs on SG1, and **SP2,** which applies other distributed firewall rules and agentless antivirus to all VMs on SG1.

Service Composer brings a key advantage to virtualized environments: it can guarantee the enforcement of security policies for current and future virtual machines, as long as they respect the criteria defined on the security groups.

Over the years, NSX Service Composer proved to be an integral part of interesting security projects such as:

- **DMZ Anywhere:** Regardless of their location on the network or physical servers, VMs identified as DMZ-worthy are automatically classified via name recognition and protected from all other networks by distributed firewall and next-generation firewall rules.
- **PCI Zone Segmentation:** Without the need to build a different virtualization cluster or network segment, VMs can be segregated from non-compliant VMs via security tags and distributed firewall rules.
- **Secure Virtual Desktop Infrastructure:** Virtual desktops can be secured against each other via distributed firewall rules while also benefitting from agentless antivirus solutions from security partners.

Identity Firewall

Identity Firewall (IDFW) provides an NSX administrator the ability to create distributed firewall rules based on Active Directory (AD) groups.

Figure 3-11 further details how IDFW works.

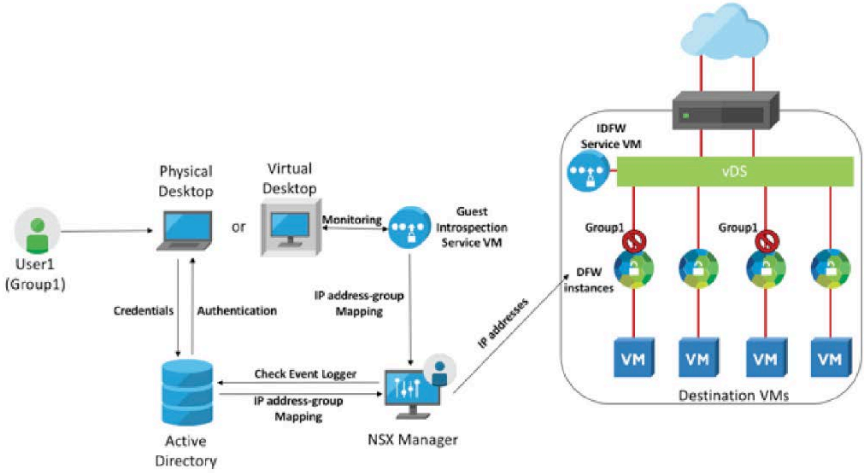


Figure 3-11 Identity Firewall in Action

Through a management connection with the AD server, an NSX administrator can use available AD groups - which appear as *Directory Groups* - in Service Composer.

The NSX Manager recognizes the IP addresses from physical and virtual machines that have logged AD users and uses these addresses on the rules and policies that are related to this group - Group1 in Figure 3-11.

IDFW uses two methods for login detection:

- **Guest introspection - only virtual desktops:** A thin agent installed on the VM forwards the information to the guest introspection service VM, which is in constant communication with the NSX Manager.
- **Active Directory Event Log Scraper - physical or virtual desktops:** NSX Manager pulls events from the AD security event log.

The VMware NSX identity firewall requires that the destination for the desktop traffic is a VM. IDFW supports the following Windows versions: 2012, 2008, 2008 R2 Active Directory servers and Windows 2012, 2008, 2008 R2, 10, 8, and 7 desktops. IDFW does not support shared desktops, terminal servers where multiple users share an IP address, or environments with NAT.

Note: Always refer to VMware documentation to verify if these features are available on the software version in use.

Micro-segmentation Defined

Micro-segmentation is the process of subdividing a network segment in order to implement the Zero Trust model on any environment.

When NSX introduced the concept of distributed firewall in 2013, many security professionals associated both words as synonyms. The assumption is reasonable because this feature leads to the creation of individual micro-segments for each one of the VMs connected to a single network segment.

With the widespread popularization of the term, and to avoid any confusion, it became paramount for VMware to formally define micro-segmentation according to its expected outcomes:

- **Distributed stateful firewalling for topology agnostic segmentation**
- **Centralized ubiquitous policy control of distributed servers**
- **Granular unit-level controls implemented by high-level policy objects**
- **Network overlay-based isolation and segmentation**
- **Policy-driven unit-level service insertion and traffic steering**

Source: https://blogs.vmware.com/networkvirtualization/2016/06/micro-segmentation-defined-nsx-securing-anywhere.html/

Besides offering specific features to achieve these outcomes (e.g., distributed firewall, NSX Manager API and GUI, service composer, logical switching, and network introspection), NSX micro-segmentation implementation also closely follows all recommendations from NIST 800-125b "Secure Virtual Network Configuration for Virtual Machine (VM) Protection" described on Section 4.4, which are:

VM-FW-R1: *In virtualized environments with VMs running delay-sensitive applications, virtual firewalls should be deployed for traffic flow control instead of physical firewalls, because in the latter case, there is latency involved in routing the virtual network traffic outside the virtualized host and back into the virtual network.*

VM-FW-R2: *In virtualized environments with VMs running I/O intensive applications, kernel-based virtual firewalls should be deployed instead of subnet-level virtual firewalls, since kernel-based virtual firewalls perform packet processing in the kernel of the hypervisor at native hardware speeds.*

VM-FW-R3: *For both subnet-level and kernel-based virtual firewalls, it is preferable if the firewall is integrated with a virtualization management platform rather than being accessible only through a standalone console. The former will enable easier provisioning of uniform firewall rules to multiple firewall instances, thus reducing the chances of configuration errors.*

VM-FW-R4: *For both subnet-level and kernel-based virtual firewalls, it is preferable that the firewall supports rules using higher-level components or abstractions (e.g., security group) in addition to the basic 5-tuple (source/destination IP address, source/destination ports, protocol).*

NSX ensures the security of physical workloads, as physical-to-virtual or virtual-to-physical communication can be enforced using distributed firewall rules. For physical-to-physical communication, NSX can provide additional security via the NSX Edge Service Gateway, as presented in the next section.

Edge Firewall

As briefly mentioned in Chapter 2, NSX Edge Services Gateways offer firewall features that are commonly known as *Edge firewall.* This feature was originally designed to enforce security on the north-south traffic between NSX virtualized networks and the physical network.

An NSX Edge can be configured as an L3 and L4 stateful firewall, with its rules processed in ascending order until there is a traffic match or the default rule is enforced.

By default, an Edge firewall blocks all incoming traffic, though it is possible to change this behavior. The Edge firewall can be configured to automatically generate firewall rules that are required for control plane communication related to features such as IPsec VPN and load balancing. This behavior that can also be changed.

Figure 3-12 portrays the configuration of Edge firewall rules.

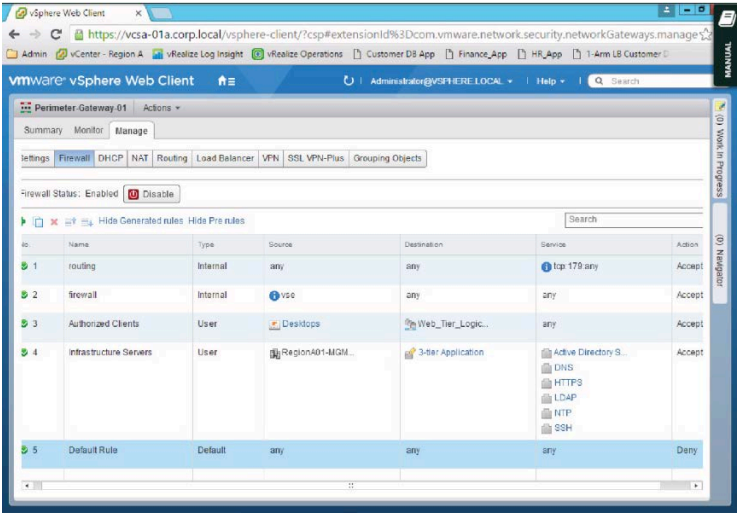


Figure 3-12 Edge Firewall Rule Configuration

In this example there are two auto-generated rules: **routing** and **firewall,** which respectively allow dynamic routing protocol communication with the NSX Edge and any traffic originated from the virtual appliance. Just before the default rule, two rules were inserted:

- **Authorized Clients:** Accepting traffic from an IP set called **Desktops** containing the IP addresses of multiple desktops to all virtual machines connected to **Web_Tier_Logical_Switch.**

- **Infrastructure Servers:** Accepting infrastructure traffic from VMs running on cluster **RegionA01-MGMT01** to security group containing all VMs that are part of the three-tier application (names have “web”, “app”, or “db” on their names).

For reference, Table 3-4 lists all parameters from an Edge firewall rule.

Table 3-4 VMware NSX Edge Firewall Rule Parameters

Rule Parameter	Description
Number	Determines the order of execution of the rule.
Name	String identifying the rule.
Rule Tag	Unique system-generated identifier for the rule
Source	Specify the traffic source described in the rule, which may include: cluster, data center, distributed port group, IP set, legacy port group, logical switch, resource pool, security group, vApp, virtual machine, vNIC group.
Destination	Specify the traffic destination described in the rule, which may include: cluster, data center, distributed port group, IP set, legacy port group, logical switch, resource pool, security group, vApp, virtual machine, vNIC group.
Service	Specifies a port protocol combination (predefined list or customizable services)
Action	Defines the rule behavior when the described traffic is identified. It can be: Accept, Deny, or Reject (sends RST message for TCP connections or ICMP messages for UDP, ICMP, and other IP communications. Also allows advanced options such as: Log, Comments, Match On (translated or original IP address), and Enable rule direction (in or out).

Note: Because an NSX Edge is a VM, it can also be used to enforce firewall rules between VLANs that are connecting physical and virtual workloads.

Security-based Virtual Private Networks

Approximately ten years before the hype surrounding Cloud and SDN, another buzzword was dominating conversations around water coolers in IT departments: *virtual private networks* (VPNs). Much like their 21st century counterparts, the term VPN conceals a series of technologies and alternatives that may not be entirely clear at first hearing.

Although all VPN implementations share a common intention - to extend a private network over a public infrastructure such as the Internet or a service provider backbone - they vary immensely according to their public network infrastructure, main objective, and protocols. And because the Internet is considered the Wild West of networking due to its gigantic number of users, security is the most usual objective for Internet-focused VPNs.

The NSX Edge Service Gateways offer several types of virtual private network technologies for access over the Internet: *IPsec VPNs* and *SSL VPN-Plus*. The next sections explore their use cases and advantages in comparison to traditional solutions based on physical appliances.

IPsec VPN with VMware NSX

Internet Protocol Security (IPsec) is actually a suite of protocols that work in tandem to authenticate and encrypt IP packets transported over a shared IP network such as the Internet. From an extensive list, the most used protocols in the IPsec suite are:

▪ **IP Authentication Header (AH):** Provides data integrity and authentication for IP packets and offers protection against replay attacks.

▪ **IP Encapsulation Security Payload (ESP):** Brings data confidentiality, authentication, integrity, and protection against replay attacks to IP packets.

▪ **Internet Key Exchange (IKE):** Establishes security associations between two IPsec peers through the negotiation of a set of algorithms and parameters that will be used to encrypt and authenticate IP packets in one direction. IKE Phase 1 validates both peers through a secure key exchange connection whereas Phase 2 establishes the communication channel for actual data plane traffic.

IPsec can be implemented in two modes: *transport mode,* where only the payload of the packet is encrypted and authenticated while the IP header is sent without any change or *tunnel mode*, where the entire packet - header plus payload - is encrypted, authenticated, and encapsulated into another IP packet.

Note: Tunnel mode is by far the most popular mode on IPsec implementations.

IPsec tunnels, which can be thought of as an overlay between two authenticated peers, can be established between networking devices (e.g., router, firewall, or VPN concentrator) or computers deploying a software IPsec client. These implementations are commonly known as *site-to-site IPsec VPNs* and *remote client IPsec VPNs.*

NSX offers IPsec site-to-site VPNs between an NSX Edge and IPsec-capable network devices installed on remote sites. In such scenarios, all remote hosts connected to this device will be able to securely access resources that are connected through L3 to the NSX Edge as if both IPsec peers were connected via a single hop connection: the IPsec tunnel.

Figure 3-13 graphically represents both a traditional IPsec VPN design via a firewall with such functionality and an NSX IPsec VPN design.

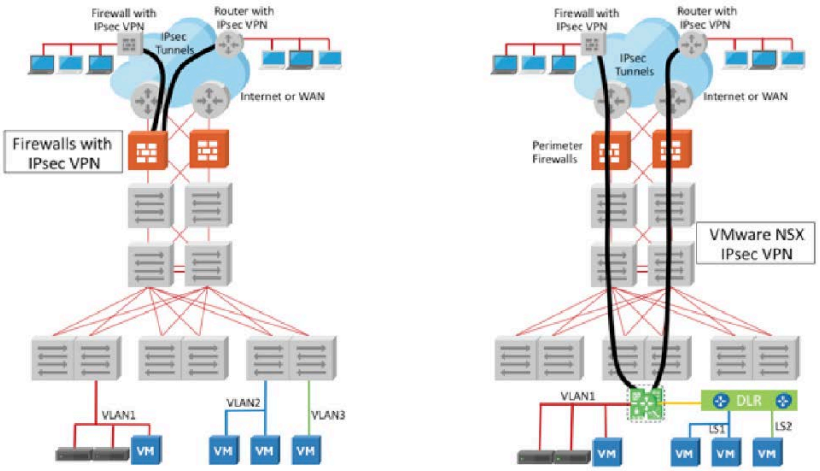


Figure 3-13 IPsec VPN Designs

On the left side of Figure 3-13, a perimeter firewall is concentrating IPsec tunnels from two remote sites. On the right, an NSX Edge is performing the same function, but in a slightly different way.

These are the main advantages of NSX IPsec VPNs over physical appliance-based implementations:

- **Simplicity:** The deployment of an NSX Edge is much simpler than the installation and configuration of a physical appliance. Physical installation requires rack accommodation, cabling, energizing, and in many cases device-by-device configuration.

- **Scalability:** It is possible to scale up (i.e., changing NSX Edge sizes) or scale out (i.e., Edges per tenant or via logical load balancing) IPsec VPNs with VMware NSX.

- **Multitenancy:** In a multi-tenant model, it is easier to dedicate an NSX Edge per tenant than to segregate rules per tenant in the perimeter firewall.

- **Manageability:** All NSX Edges can be controlled by a single NSX Manager instance and monitored by a logging server such as VMware vRealize® Log Insight™, which offers a special dashboard for this NSX Edge service.

- **Traffic control:** If it is not restricted by rules configured on the perimeter firewall, a remote host can gain access to unauthorized internal resources such as servers or network devices. In the case of VMware NSX, a simple rule in the perimeter firewalls permitting

IPsec protocols between the remote site and the NSX Edge – and denying everything else - will limit remote access to the internal networks connected to this NSX Edge's virtual topology.

Note: VMware NSX supports the following IPsec features: pre-shared keys, digital certificate authentication, Phase 1 Main Mode, Phase 2 Quick Mode, Diffie-Hellman Groups 2 and 5, Perfect Forward Secrecy (PFS), static routing over IPsec tunnels, Rivest-Shamir-Adleman (RSA) and Digital Signature Algorithm (DSA) with 2048 and 3072 bytes keys, Triple Data Encryption Standard (3DES), Advanced Encryption Standard (with 192 and 256 bits), and Advanced Encryption Standard with Galois/Counter Mode (AES-GCM). As always, it is highly recommended to review VMware's documentation to verify the features available in any specific NSX version.

SSL VPN-Plus with VMware NSX

Although IPsec can provide VPN access directly to desktops, notebooks, and mobile devices, the installation, configuration, and support of software agents on numerous heterogeneous devices is a considerable operational challenge. Issues such as compatibility, MTU adjustment, and secure delivery of IPsec parameters have encouraged the creation of an alternative method for remote client VPN scenarios.

In the early-2000s, *Secure Socket Layer (SSL) VPNs* were introduced. Their main purpose is to provide secure access to remote hosts without requiring the interactive installation of specialized client software on these computers. Different from IPsec VPN, an SSL VPN connection does not encapsulate IP packets into other IP packets; instead, as its name infers, it uses an ordinary SSL connection from a standard web browser.

Such simplicity was quickly translated into popularity, and it was only natural for VMware NSX to offer such VPN option among its portfolio of features. Formally known as *SSL VPN-Plus* and also implemented on NSX Edges, this functionality provides secure remote access through two methods:

- **Web-access mode:** A remote user is authenticated and granted access to private resources listed in a customizable web portal.
- **Network-access mode:** A remote user downloads a pre-configured SSL VPN-plus client from a web portal. With this client, remote users can access resources on the private network in a similar way as hosts connected through a IPsec VPN remote client, but without the logistical hurdles related to its maintenance.

Figures 3-14 and 3-15 illustrate the SSL VPN-Plus web portal login page and the SSL VPN-plus client download page, respectively.

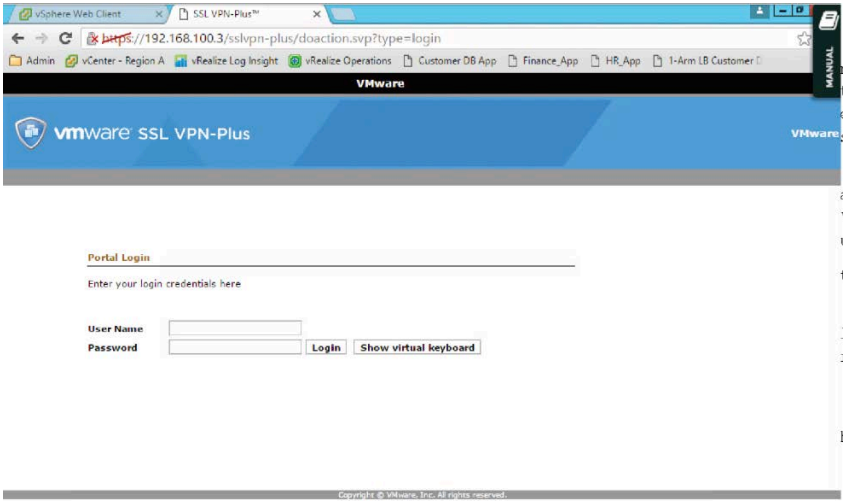


Figure 3-14 SSL VPN-Plus Web Portal Login Page

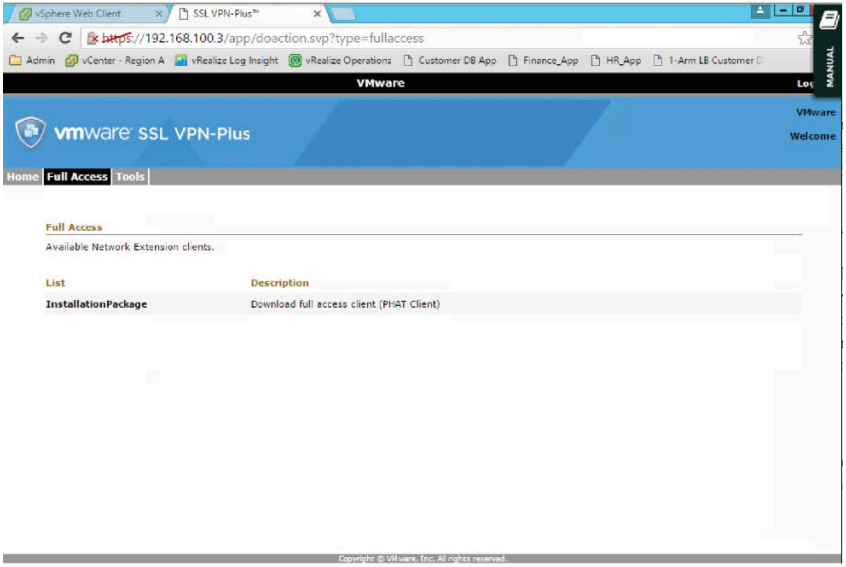


Figure 3-15 SSL VPN-Plus Installation Package Page

Much like NSX IPsec VPNs, SSL VPN-Plus presents multiple advantages over traditional SSL VPN solutions deployed through physical appliances. To avoid repetition, please review the advantages (i.e., simplicity, scalability, multitenancy, manageability, and traffic control) listed on the "IPsec VPNs with VMware NSX" section.

Note: VMware NSX SSL VPN-Plus supports user authentication via local user database, Active Directory (AD), AD over SSL, Lightweight Directory Access Protocol (LDAP), LDAP over SSL, Remote Authentication Dial-In User Service (RADIUS), and RSA SecurID. It also deploys TCP optimization to eliminate performance degradation due to TCP-over-TCP behavior. The SSL VPN-Plus client is compatible with: Windows XP and newer; Mac OS X Tiger, Leopard, Snow Leopard, Lion, Mountain Lion, and Mavericks; and Linux. SSL VPN-Plus in web access mode was deprecated in version 6.2.4. Refer to the latest VMware documentation to verify if version in use supports these functionalities.

Other Interesting VMware NSX Security Features

Even though micro-segmentation-related features dominate discussions around NSX in security circles, the platform has more to offer in terms of protection features. Details on two of those topics will be discussed before closing this chapter.

SpoofGuard

One of the oldest ways to try to bypass firewall policies is changing a computer's IP address. A more destructive attack can be achieved if the hacker spoofs the subnet's default gateway IP address in such computer, creating a *blackhole effect* in this network, where other hosts may try to use the compromised computer as its default gateway.

To avoid such unpleasant situations in virtualized environments, NSX offers a feature called *SpoofGuard.* This feature collects the IP addresses of all guest virtual machines, via VMware Tools, DHCP snooping, or ARP snooping, as well as their MAC addresses via VM definition files, and blocks traffic if a change on such parameters is noticed.

SpoofGuard is especially useful in public and private cloud implementations where many tenants have full administrative access to their virtual machine operating systems.

Endpoint Monitoring and Application Rule Manager

Endpoint Monitoring profiles applications that are running inside a VM guest operating system, providing visibility into specific application processes and their associated network connections. *Application Rule Manager* analyzes the connections from selected VMs to suggest security groups and firewall rules.

Through such end-to-end analysis of applications and their communication, Endpoint Monitoring and Application Rule Manager further simplify the adoption of micro-segmentation on virtualized environments.

Note: Endpoint Monitoring leverages guest introspection for application process deep analysis. It supports Windows-based virtual machines with VMware Tools installed and running.

What Did This Chapter Achieve?

Figure 3-16 depicts a single network topology comprising the main concepts discussed in this chapter.

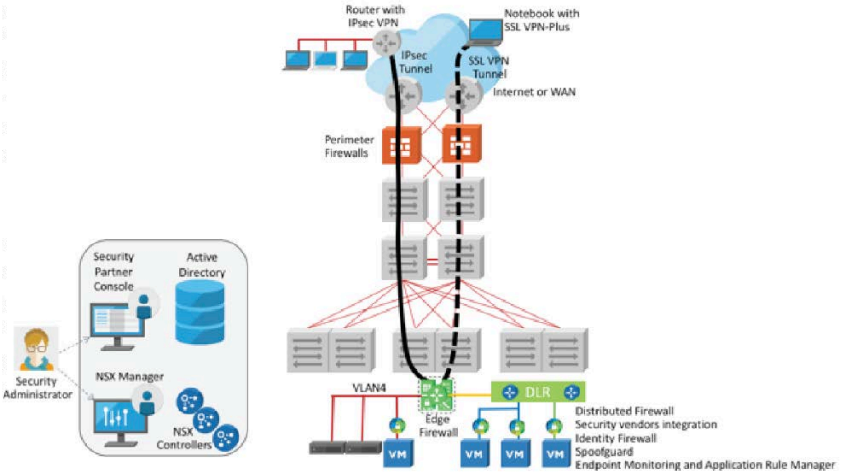


Figure 3-16 VMware NSX Security Features

In summary, they are:

- **Distributed Firewall:** Enables the enforcement of security rules based on logical criteria (e.g., virtual machine name, security tags, among others) in virtual machines connected to standard VLANs or VMware NSX logical switches. The distributed firewall is a kernel-level module that protects each individual virtual machine vNIC.
- **Security Partner Integration:** Allows the integration of third-party security solutions without network reconfiguration. It uses network introspection (e.g., NGFW, IPS, and IDS) and guest introspection (e.g., antivirus and vulnerability management).
- **Identity Firewall:** Based on the NSX distributed firewall and leveraging guest introspection, this functionality enables the enforcement of firewall rules based on Active Directory groups.
- **Edge Firewall:** L3-4 firewall running on an NSX Edge, capable of enforcing traffic based on logical criteria (e.g., name, logical switch, among others) for both VMs and physical workloads.
- **IPsec VPN:** Creates a secure site-to-site connection between an NSX Edge and a remote router or firewall deploying IPsec.
- **SSL VPN-Plus:** Creates a secure remote connection between an NSX Edge and a host based on the SSL protocol.
- **SpoofGuard:** Collects virtual machines IP and MAC addresses, then automatically blocks traffic if an address change is detected.
- **Endpoint Monitoring:** Uncovers the relationship between application processes and their associated network connections among multiple virtual machines.
- **Application Rule Manager:** Analyzes the connections from selected VMs to suggest NSX security groups and firewall rules.

Virtual Networks, Real Extensions

> *"Some people don't like change, but you need to embrace change if the alternative is disaster."*
> *Elon Musk*

There is little question on how important data centers have become for organizations all over the world. Within this secure vault of applications and data, corporations manage sales, run processes, build customer relationships, and safely store strategic information for their future.

Regardless of their industry, the overwhelming majority of companies possess at least two data center sites to avoid risking their information assets in the case of a catastrophic failure. These redundant structures usually contain a very similar set of software and hardware, and are often separated by a reasonable distance to avoid fate sharing in the case of natural disasters, major accidents, complete site failure, or more trivial events such as power outages.

Notwithstanding, having two of everything does not guarantee business continuance against such misfortunes. Because there is an intimate relationship between infrastructure and applications, multiple organizations frequently plan, test, and revaluate disaster recovery plans to expeditiously recover their IT systems after a disastrous event.

During the first decade of the 21st century, revolutionary technologies such *server virtualization* helped to simplify and streamline the huge number of procedures for different systems in disaster recovery plans. Networking has remained a challenging subject for most distributed data center designs. This chapter intends to explain these challenges while exploring the evolution of disaster recovery strategies during the last few decades. With this context in place, the chapter will then demonstrate how NSX can reduce complexity on multisite data center network designs with both agility and security.

The Evolution of Secondary Sites

Information technology turned into a strategic business advantage in the 1960s when mainframes became a commercially affordable resource for corporations outside of research and governmental circles. From this period onward, these organizations started to harvest great results that were unimaginable before, while also experiencing the bitter taste of not achieving them due system malfunctions and disasters, both man-made and natural.

A working application has always relied on the collaboration of different areas (e.g., facilities, computing, storage, networking, software development among others). IT teams quickly saw that just deploying high availability for each system could be both costly and ineffective as an application recovery strategy. These teams realized that a detailed *disaster recovery plan* was necessary to correctly respond to a total loss or major failure of an IT system through coordinated procedures involving all stakeholders.

Moreover, it was perceived that this plan should be also part of a bigger *business continuity plan* that defined management processes to eliminate, or at least minimize, business losses according to each identified threat and known impact.

To balance effectiveness and cost control, two key variables should be defined as the starting point of any of these plans:

- **Recovery Point Objective (RPO):** Maximum planned time interval in which application data can be lost before a major incident.
- **Recovery Time Objective (RTO):** Maximum amount of time after a major incident after which an application must be fully operational.

The closer RPO and RTO get to zero, the more complex and expensive are the solutions required to achieve them.

In the context of IT systems, the use of a single site was immediately characterized as a major risk. Organizations started to consider the inclusion of a *secondary site* to their spending budget in order to lessen the probability of major disasters occurring at both locations at the same time.

Note: Many organizations are forced to maintain two or more data center sites due to business requirements and compliance regulations.

The *how* of an organization's use of its secondary site effectively has evolved continuously according to business requirement changes and technology innovation. The next sections delineate a simplified historical narrative to epitomize the disaster recovery strategies for distributed data centers that were developed over the last five decades.

The Cold Site

During the late 1960s and throughout the 1970s, organizations acquired mainframe systems and installed them in pristine new *computer rooms*.

In that era, applications were not real-time. They were based on *batch processing*, where periodic occasions were set in advance to allow data input and results gathering. Both RPO and RTO desired for these systems could reach weeks, depending on the application.

For this reason, the large majority of organizations preferred not to invest resources in additional computer rooms; instead, they hired the service of a *"cold site"* from providers such as IBM and SunGard.

This disaster recovery strategy is illustrated in Figure 4-1.

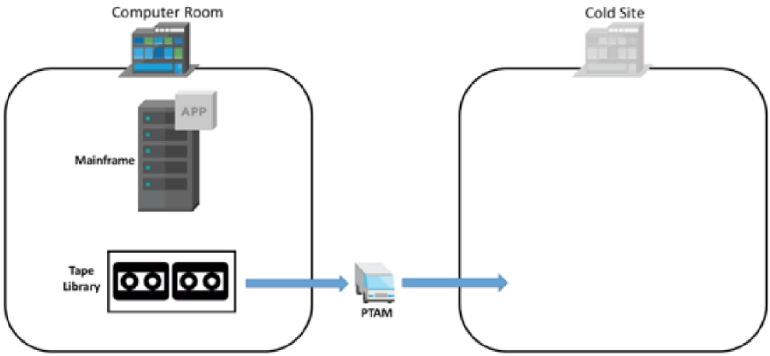


Figure 4-1 The Cold Site

As Figure 4-1 shows, data was stored on tapes during this period. Recovery plans were focused on data backup and restore.

According to the user agreement, the chosen service provider was expected to offer a backup site that was prepared to receive all necessary hardware - mainframe and tape drives - to run the batch process.

After a computer malfunction, working hardware had to be installed at the cold site while software, both operating system and application, was manually installed to match the mainframe characteristics from the original customer setup.

Data transport was carried out through the physical shipment of tapes with saved data to the backup site. Amusingly, this primitive networking approach was formally called Pick-up Truck Access Method (PTAM).

At the time of recovery, data was loaded and restored into recently installed tape libraries at the cold site to run the customer desired batch process. With such long recovery objectives, there was also enough time to fix or replace the customer system until the next batch process.

The Warm Site

The warm site was a natural variation of the cold site, and it was more commonly used in the 1980s. This strategy acknowledged that many batch applications simply had narrower unavailability windows for business-related reasons. With the purpose of pursuing an RPO and RTO of days, the same service providers offered what is represented in Figure 4-2.

Figure 4-2 clarifies the main differences between cold and warm sites. In order to shorten the RTO, the latter offered hardware and software

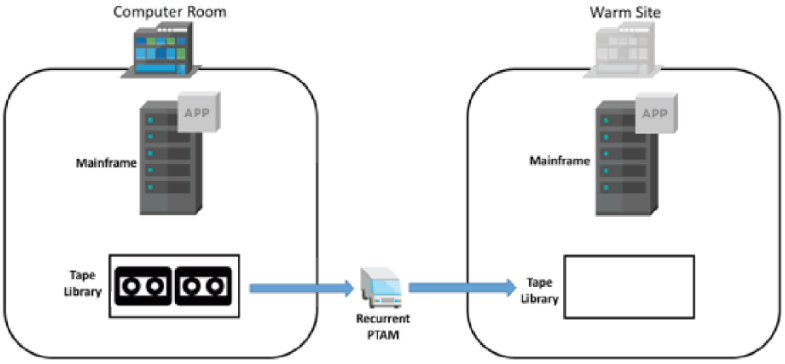


Figure 4-2 Warm Backup Site

already installed at the warm site. At the same time, backup tapes were transported more frequently to the backup facility in order to decrease the RPO and accelerate the manual data restoration in case something happened to the primary computer room.

Because a mainframe service provider offering warm site services should keep working systems for many customers concurrently, they usually relied on the mainframe virtualization technologies that were briefly mentioned on Chapter 1, "What Is Network Virtualization?". Alternatives, some organizations avoided the higher operational costs of such a service by building and maintaining their own warm sites.

The Hot-Backup Site

The 1990s witnessed the birth of online electronic business through the Internet boom. With application clients sending web-based requests and expecting results in real time, IT teams were forced to radically change their disaster recovery strategy.

With the sum of RPO and RTO tightened to a few hours, most companies started deploying *hot sites*, which were promptly ready to overcome major incidents at the primary data center site.

Figure 4-3 graphically portrays the main principles of the hot site, which often were refurbished computer rooms populated with midrange systems and microcomputer-based servers.

As Figure 4-3 explains, the motto of a hot site could be summarized in

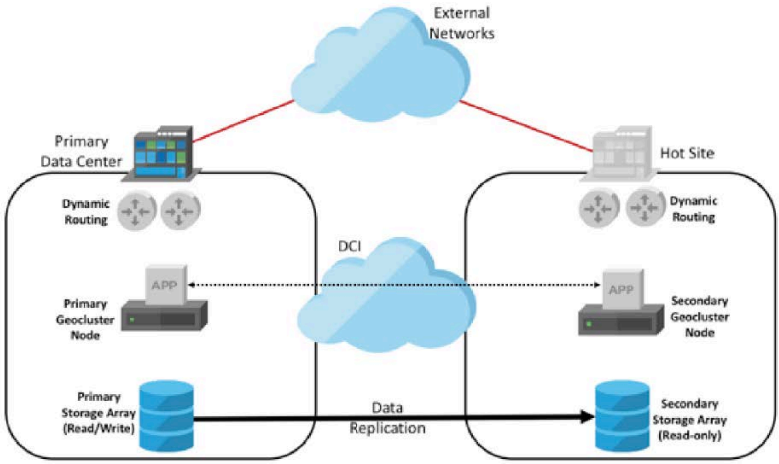


Figure 4-3 The Hot Site

three words: ready to go. Manual tasks were minimal and, preferably, logical rather than physical in every single area of IT expertise.

During that time, storage arrays were the system of choice to stock application data. Their *data replication* capabilities were deployed to guarantee that a predefined RPO was respected via application data availability at both sites. Depending on the aggressiveness of this metric, these arrays could deploy one of two types of replication:

- **Synchronous replication:** Storage device on the primary site only confirms a server-requested data change, also known as *write*, if this change was fully confirmed to be copied to the secondary site, which traditionally only supports data read operations. In order to minimize application latency, sites deploying synchronous replication are usually located no further apart than 100 kilometers.

- **Asynchronous replication:** Storage device on the primary site copies data to the secondary site in periods shorter than the RPO. Because applications servers can save data without the need to wait for any storage replication process, data centers using asynchronous replication do not influence on application performance.

To avoid application downtime due to processing-related operations, *geographic clusters* (geoclusters) were conceived and implemented during this period. A geocluster is a system composed of a set of servers, also called *nodes*, that are running the same application at distributed sites. If a failure is detected on the primary node, one of the secondary nodes will resume the application operations.

Generically speaking, geocluster nodes leverage the storage replication strategy to guarantee that the application, regardless of the site where it may run, will access the same data set. A secondary node verifies whether the active node is operational and is prepared to react properly if a failure is detected. In a failure case, the secondary node activates itself and breaks the replication process to start writing on the storage array located in the hot site.

Note: There are multiple examples of geographic cluster technologies such as Oracle Clusterware, Veritas InfoScale Availability, and Microsoft Cluster Services. Although their internal implementations are widely different, the generic model presented in this writing only intends to explain the dependency of such technologies over storage and networking designs.

Through real-time applications, networking gained relevance it lacked in prior computing architectures. As an illustration, dynamic routing to external networks was heavily developed during this period, in order to ensure that application users could always reach active nodes in the

event of a failure on a network device.

Note: Future sections in this chapter will explain edge routing techniques for disaster recovery strategies in more detail.

Besides client traffic accessing the application nodes via Internet or any other external network, many data center architects leveraged an extra *data center interconnect* network to transport relatively new types of traffic such as storage replication via IP and inter-node communication (e.g., geocluster "heartbeats" and "in-memory" application state information).

The Virtualized Hot Site

During the 2000s, it was only natural that more heavily used applications required tighter RPOs and RTOs measured in minutes. At the same time, data centers were changing their atomic unit to the concept of the virtual machine. With VMware and other hypervisor vendors providing the capability of hosting multiple virtual servers on a single x86 physical server, many data center architects achieved an unprecedented higher utilization of resources.

This technology enabled additional benefits. Because virtual machines were basically files, disaster recovery strategies could be drastically simplified as server virtualization normalized the considerable number of clustering technologies that were used for distinct applications.

Figure 4-4 clarifies this statement.

As hinted at in the diagram, server virtualization clusters could greatly

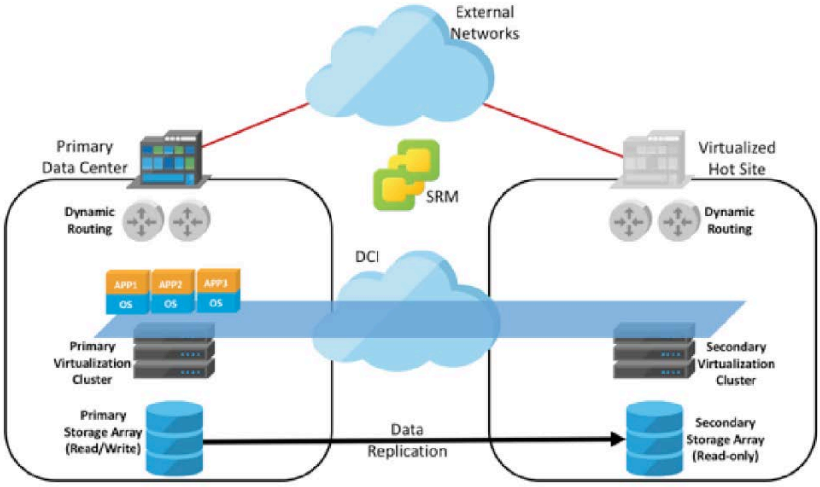


Figure 4-4 The Virtualized Hot Site

benefit from a *disaster recovery orchestration tool* such as VMware Site Recovery Manager™ (SRM), to promptly react to a major failure in the primary site. Such tools can coordinate the following actions:

- Interrupting storage replication in order to make the storage array in the secondary site available for data access (i.e., writes).
- Reigniting failed virtual machines in the virtualization cluster at secondary sites as soon as possible, leveraging the fact that their associated files were replicated to the secondary site when the replication process was still active.
- Performing the necessary steps to facilitate connectivity to the born-again VMs.

Server virtualization did not require applications to be modified in order to take advantage of the new benefits it introduced, which significantly streamlined disaster recovery plans.

Up to this period, all disaster recovery strategies were based on *active-standby* sites, where a major failure would switch all application components to the secondary site. Although it may be viewed as a bit radical, it certainly simplified disaster recovery plans.

The Active Site

Change remained the only constant in recovery site architectures. With the financial crisis of 2008, IT budgets were more intensely scrutinized. Many organizations began to evaluate the true advantage of investing in a whole data center facility used only for a few hours per year during disaster recovery tests and actual major failures.

As a way to increase the efficiency of existing resources and avoid acquisitions to support systems expansions, many data center architects kickstarted projects with active workloads on the secondary site, as Figure 4-5 exemplifies.

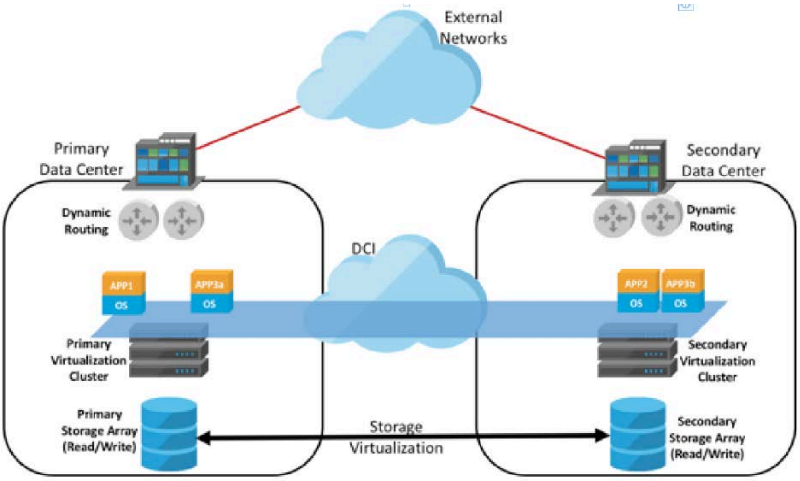


Figure 4-5 Active-Active Sites

There are two basic approaches to use distributed data center sites in an active-active strategy:

- Distributing active applications among the active sites and using the alternate site in the case of a disaster. Applications APP1 and APP2 are examples of this strategy.
- Having the same application simultaneously running with active instances on both sites (e.g., APP3).

Such desires were not 100% achievable until they were supported by technology innovation. For example, storage virtualization enabled *bidirectional replication*, which allowed writes on storage arrays at both active sites with automatic replication to the other site.

Another impressive innovation of the 2010s, server virtualization granted long distance live migration of virtual machines between sites, also known as long distance vMotion. While this type of vMotion is not considered a disaster recovery tool because a VM cannot be migrated *after* a major failure on its site, it can be perceived as a *disaster avoidance* technique that allows the transfer of active applications before an anticipated disaster or planned maintenance.

Active-active sites are usually deployed during site migrations - in which a new data center will replace an old one - or data center expansion when a site has run out of capacity.

In addition, data center architects must be aware that the utilization of both sites should not be steadily over fifty percent to avoid the situation where a major site failure would impact in overall performance in the remaining site. For this reason, many financial corporations deploy a mixed strategy with two active-active sites and a third one that is used exclusively for disaster recovery.

Networking Challenges in Active-Standby Sites

In active-standby data center architectures, applications are expected to run on only one site at a time. Consequently, during normal operating conditions, production network traffic occurs at the active site while the standby site's networking infrastructure handles marginal management and "heartbeat" communication.

If a disaster hits, the network infrastructure in the standby site should be capable of supporting all production traffic as promptly as possible. Hopefully, all applications should continue to be protected from attacks and vulnerabilities.

Although such network design may seem simple at first glance, there are a considerable number of tasks that are necessary to prepare both sites for an active-standby operation. Figure 4-6 provides background for such a discussion.

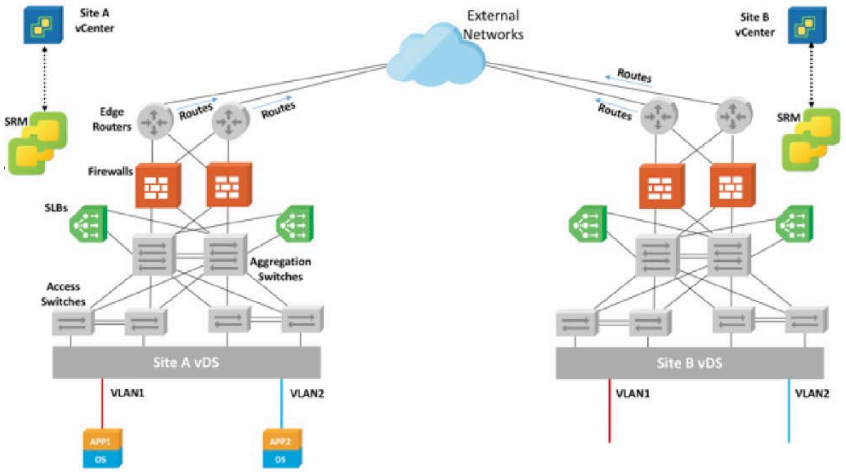


Figure 4-6 Active-Standby Network Topology

In this scenario, a disaster recovery orchestration tool - VMware SRM, in this case - is used to automate VM switchover to site B if a major failure is declared at site A. From a networking perspective, SRM could change the IP addresses of the VMs when they are bring recovered in site B. Such operation could generate unnecessary complexity because the original IP addresses are registered on other systems such as DNS, firewall rules, and even on other applications. For such reason, most data center architects avoid IP readdressing during disaster recovery operations.

Under such circumstances, for every new application that must be included on the active-standby scenario shown in Figure 4-6, a network operator will need to perform tasks like the ones described in Table 4-1.

Table 4-1 Network Operations for Applications on Active-Standby Sites

Type of Operation	Configured Devices	Brief Description
Layer 2	All aggregation and access switches, firewalls, SLBs, and virtual distributed switches on both sites	Depending on the logical topology, one or more VLANs should be configured on all of these devices and added to each trunk. On the vDS, these VLANs are offered to the VMs as distributed port groups (dPGs).
Layer 3	Firewalls, SLBs, aggregation switches, and edge routers on both sites	IP address configuration on firewall, SLBs including the application VIP, or aggregation switches to provide default gateway for the servers. In the case the firewalls do not support dynamic routing protocol, the application subnet should be redistributed into the dynamic routing protocol configuration. Because the Edge routers at site B should remain active, they should advertise the new application route with a higher cost than site A to avoid attracting client traffic.
Security	Firewalls on both sites	Rule configuration for the new application.
Application Availability	SLBs on both sites	Configure server pool, monitors, load balance algorithm, and VIPs installed on different subnets.

As Table 4-1 outlines, these tasks are not trivial in number or complexity. They require varied expertise and care when accessing all management points – such as switches, routers, firewalls, SLBs, and vCenters - at both sites.

An IT rule of thumb states that complexity means fragility. Numerous operations can easily be translated in human configuration errors which are overlooked during the usually stressful times following major failures. SRM and other disaster recovery tools allow the implementation of tests to verify how correct these network configurations really are.

VMware NSX for Active-Standby Sites

The VMware NSX kernel-based objects explained in Chapter 2, "Rethinking Network Constructs", such as logical switches, distributed logical routers, and distributed firewalls, were originally designed to work within the scope of a single VMware vCenter Server. While some corporations may happily use a single vCenter-managed virtualization cluster extended over two distinct data center sites, such design may not satisfy more stringent disaster recovery plans for one reason: the vCenter Server itself is a key component of the recovery process for VMs at the standby site, therefore it is considered a good practice to deploy at least one vCenter at each site.

VMware NSX offers a special design called *Cross-vCenter* that is ideally focused on multisite data centers. Cross-vCenter allows the creation of *universal* network constructs that can be instantiated simultaneously on multiple vCenter Servers.

Figure 4-7 depicts an example of a VMware NSX Cross-vCenter implementation at two sites.

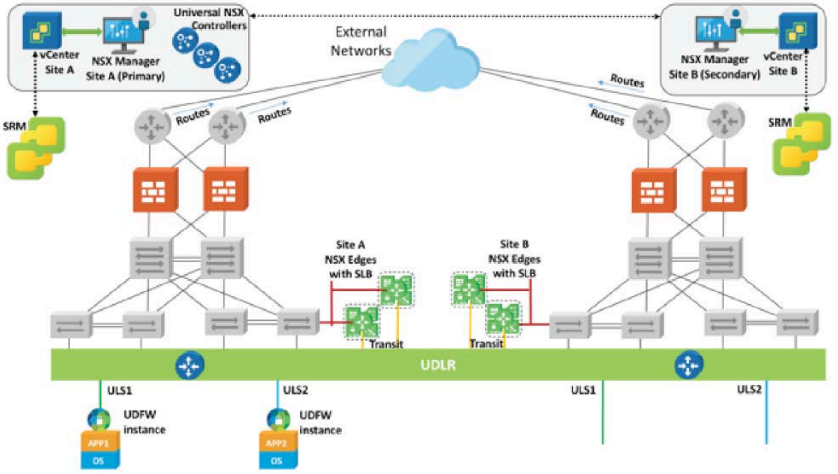


Figure 4-7 VMware Cross-vCenter Topology

Although Figure 4-7 only shows two sites, Cross-vCenter controls multiple synchronized NSX Manager-vCenter pairs that can be distributed over multiple data center sites.

Note: Please refer to NSX documentation for complete information on Cross-vCenter scalability limits. VMware NSX supports up to eight NSX Manager-vCenter pairs on a single Cross-vCenter configuration.

One of the NSX Managers is assigned the role of *primary NSX Manager* while all the others are configured as *secondary NSX Managers*. Through such arrangement and a *universal NSX Controller cluster*, an administrator can create the following NSX *universal* objects via primary NSX Manager:

- **Universal Logical Switches (ULS):** Layer 2 broadcast domains that span multiple vCenter domains that may be running on different sites.
- **Universal Distributed Logical Router (UDLR):** Distributed router extended over hosts that belong to multiple vCenter domains that may be running on different sites. Similarly to a standard distributed logical router, the UDLR can route packets between different *universal* logical switches within hypervisor hosts.
- **Universal Distributed Firewall (UDFW):** Enables centralized management of distributed firewall rules that will be enforced on VM traffic under multiple vCenter domains.

Note: The primary NSX Manager provisions all three universal NSX Controllers to handle the synchronization of VTEP, MAC, and ARP information among all hosts that belong to a *universal transport zone (UTZ)*. The UTZ defines the clusters that should implement these universal objects.

The location of the universal NSX Controllers at a single site *does not* represent a single point of failure, because these controllers are VMs that can be re-deployed as other VMs after a site switchover. Moreover, NSX has a feature called *Controller Disconnected Operation (CDO)* that allows normal data plane operations until its control plane is fully recovered.

Considering all these conditions, a new application being provisioned in an active-standby data center environment requires the configuration tasks like the ones that are listed on Table 4-2.

Table 4-2 VMware NSX Cross-vCenter Operations for New Applications Insertion on Active-Standby Sites

Type of Operation	Configured Devices	Brief Description
Layer 2	**Primary NSX Manager**	One or more universal logical switches should be created once to be replicated as distributed port groups on all synchronized vCenters.
Layer 3	**Primary NSX Manager**	UDLR interfaces are created to serve as default gateways for the VMS connected to the corresponding universal logical switches. The assigned subnets can be advertised to the physical network through dynamic routing protocols such as BGP via local NSX Edges at each site.
Security	**Primary NSX Manager**	UDFW rules can be created using flexible parameters such as IP sets, VM name and universal security tags.
Application Availability	**Primary and Secondary NSX Managers**	Logical load balancing can be deployed on existing NSX Edges or added as one-armed NSX Edges connected to a universal logical switch. Figure 4-7 leverages the former method.

Compare these procedures with the ones presented in Table 4-1. This demonstrates the greatest value of NSX in active-standby scenarios: operational simplicity and consequent reliability through management centralization.

With all these objects automatically provisioned at site B, a disaster recovery switchover can be greatly condensed as Figure 4-8 illustrates.

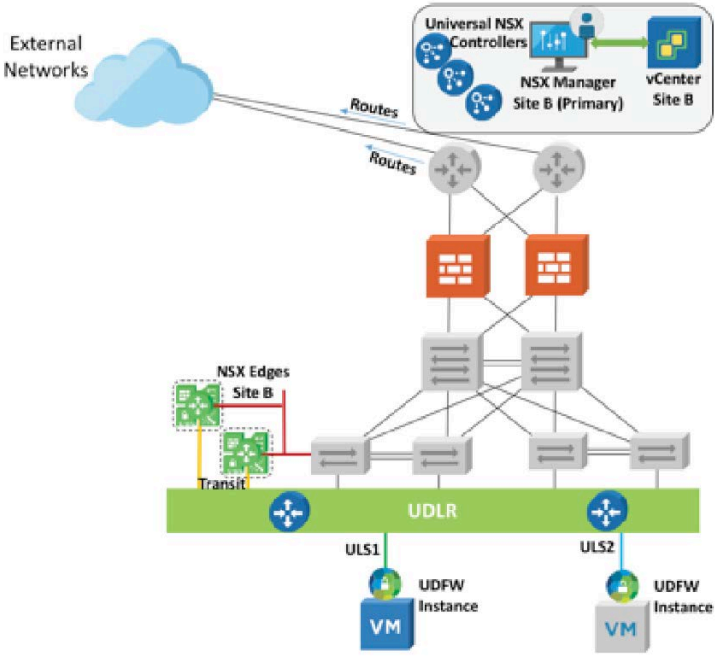


Figure 4-8 VMware Cross-vCenter After Major Failure

As the figure highlights:

- Routes to the virtual machines and SLB VIPs continue to be advertised to the external networks. Because site A routes do not exist, ingress traffic is naturally attracted to site B.
- SRM resurrects application VMs at site B. Until the universal controller cluster is rebuilt – either automatically via VMware vRealize® Orchestrator™ or manually - NSX leverages CDO mode to forward traffic on all defined universal logical switches and universal distributed logical routers.
- UDFW rules are already part of the virtual machines vNICs, from the instant they start to run at site B.
- NSX logical load balancing parameters are preconfigured and ready to receive traffic.

NSX can be used to test a recovery plan with the creation of logical networks without any disruption to the production environment.

Networking Challenges in Active-Active Sites

As discussed in section "The Active Site", an active-active strategy intrinsically means that applications are expected to run at more than one data center location. From a networking perspective, preparation should be done to support the following scenarios:

- **Geoclusters:** Different application instances running simultaneously on both sites
- **Disaster avoidance:** Live migration of VMs between both sites
- **Disaster recovery:** Per-application switchover

All of these situations invariably require communication between hosts and network devices that belong to the same IP subnet but are not in the same location, because:

1. Distributed application instances should talk to each other to verify status, synchronize application state, and leverage network services from the other site in case of a failure.

2. Migrated VMs should continue to communicate with their former subnet neighbors from the older site.

3. Applications that were switched to a secondary site due to a disaster condition may still have dependencies with other applications and services that were not migrated and with whom they may share IP subnet addresses.

All of these cases go against one of the oldest dogmas of networking: "*Thou shalt not extend subnets between different data center sites"*. For some decades, this doctrine intended to reduce VLANs to their smallest possible scope, isolate independent sites from remote network instabilities, avoid collateral effects in inter-site traffic, and decrease complex network configurations.

Each of these challenges will be addressed in the next sections to further highlight the benefits of VMware NSX for active-active data center architectures.

Spanning Tree Protocol and Failure Domains

Although *flooding* and *broadcast* are required for IP communication in Ethernet-based networks, these multi-destination frames can easily generate *loops* that can threaten all applications in a data center network.

To avoid these situations, Ethernet networks implement by default one of the versions of the *Spanning Tree Protocol (STP).* Via exchange of *Bridge Protocol Data Unit (BPDU)* frames, Ethernet switches can block traffic in chosen ports to form a loop-free logical topology called *spanning tree.*

This process is called *convergence*, and it lasts until a switch is elected to be the *root* of the spanning tree while all the other switches become tree branches in this logical topology. If an active connection fails, or if there is a change in such topology, a new spanning tree is calculated and a new convergence occurs.

Much was done to develop STP to provide faster convergence with IEEE 802.1w and independent spanning tree instances per VLAN group with IEEE 802.1s. A spanning tree primarily defines a *failure domain* that submits all participant switches to a probable convergence at any time.

Adding support to the aforementioned network dogma, a spanning tree instance spread across distributed data center sites presents even more formidable challenges, such as:

- **Scalability:** *STP diameter* can be defined as the maximum number of hops between two switches on a spanning tree. When data centers are connected through L2, this diameter can easily surpass the IEEE-recommended value of 7, causing unexpected results.

- **Isolation:** When a spanning tree exists over multiple sites, the root switch must naturally be located at one of them. If there is a topology change, all switches on all sites may need to converge again, putting all data center sites within a single failure domain.

Many organizations with reasonable access to optical fiber connections tend to think that a simple direct connection between switches is more than enough to extend VLANs between two sites. Motivated by the creation of multi-chassis link aggregation technologies, these organizations followed a best practice that is summarized in Figure 4-9.

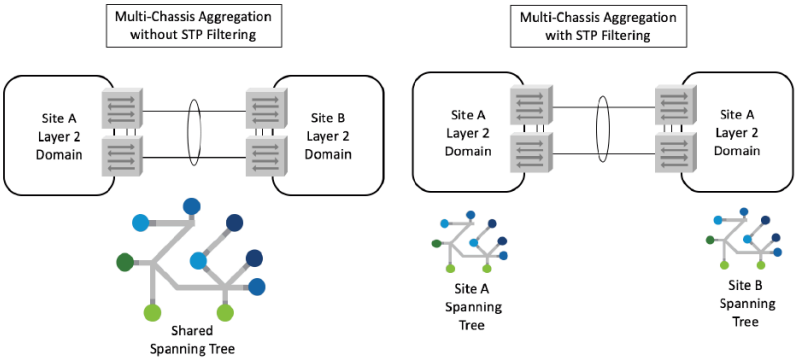


Figure 4-9 Multi-chassis Link Aggregation and BPDU Filtering

On the left side of Figure 4-9, two sites are connected without any special care related to STP. Because STP is running on the multi-chassis link, both data centers share the same spanning tree. This means that they share the same failure domain since any convergence or STP instability may create an outage at both sites.

The right side of Figure 4-9 shows what happens if *BPDU filtering* is implemented on the DCI links. In this case, because no BPDU will be allowed to traverse, each site will have its own spanning tree, thus will not be part of a single failure domain.

While this practice works in this scenario, further issues may arise when this methodology is applied to more than two data center sites, as Figure 4-10 demonstrates.

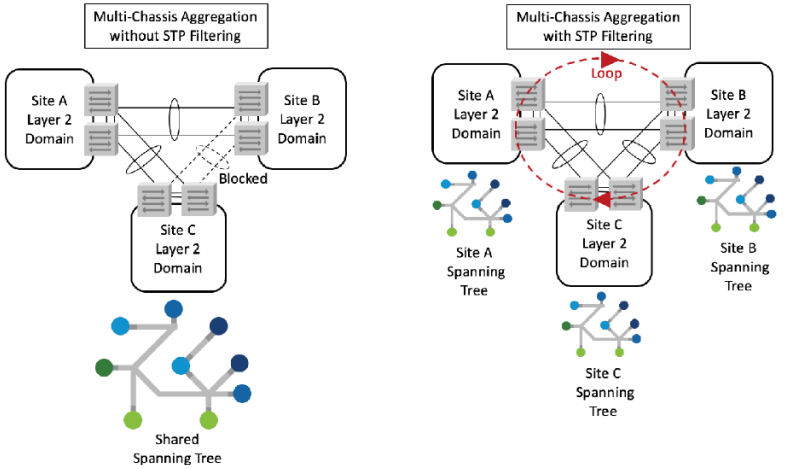


Figure 4-10 STP Filtering in Multisite

On left side of Figure 4-10, observe what is expected to happen if no special STP configuration is performed in the interconnection between sites: an enormous spanning tree generates a single failure domain for all three sites. Moreover, due to fact that only one site would host the STP root - site A in the example - all L3 DCI traffic would have to traverse it, thus forming a *hub-and-spoke* topology that disables many expensive inter-site links.

Unlike Figure 4-9, BPDU filtering would not enhance a multisite scenario for one reason: all inter-site links are now active, and without the action of STP, a loop could disrupt connectivity on all three sites!

Accordingly, the network industry has reacted using two different technologies to achieve L2 extension for multisite data centers.

These solutions are portrayed in Figure 4-11.

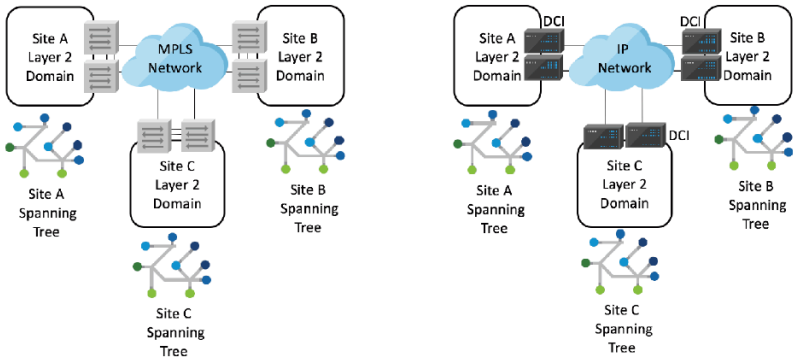


Figure 4-11 Multisite Layer 2 Extension Technologies

Figure 4-11 depicts two types of L2 extension technologies that do not merge spanning trees from different sites and do not generate loops in the DCI network. They are:

- **MPLS-based:** Technologies such as *Virtual Private LAN Service (VPLS)* and *Ethernet VPN (EVPN)* offer L2 extension between multiple sites using different methods - pseudo-wires and MP-BGP, respectively. Whereas these options are robust, they require the use of an MPLS backbone, which can be too costly for some organizations.

- **IP-based:** These solutions leverage overlay technologies - which may be proprietary - to allow the transport of Ethernet frames over an IP-based DCI network. Although these solutions may use simpler and cheaper IP networks, they usually require dedicated specific devices and an MTU increase on the IP network to support the transport of the overlay IP packets.

Layer 2 Extension and Traffic Behavior

Layer 2 extensions for active-active sites may also bring unintended effects to network traffic that are both hard to detect and troubleshoot. Figure 4-12 offers details on two such issues.

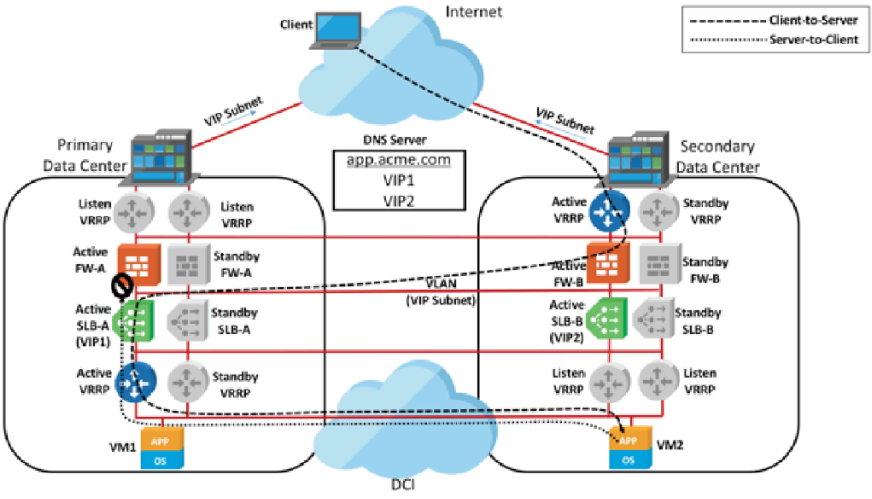


Figure 4-12 Tromboning and Traffic Asymmetry

To facilitate the visualization of the traffic effects illustrated in Figure 4-12, the following sequence of events explains each stage of the client-to-server communication.

1. As a hypothesis, VIP1 on SLB-A was selected from two entries to reach domain app.acme.com on a DNS server. The other candidate was VIP2 on SLB-B.

2. Both VIP1 and VIP2 belong to the same IP subnet hosted on an extended VLAN and that is advertised via BGP to the Internet. Depending on how the BGP is configured on edge routers from sites A and B, some client connections, or some fraction of packets, may reach site B.

3. Site B edge routers point to active FW-B due to static or dynamic routing and deliver packets to traffic destination VIP1 on SLB-A.

4. Following its configured load-balancing algorithm, SLB-A selects VM2 at site B and performs SNAT to guarantee server traffic returns to it.

5. Because SLB-A is not deploying dynamic routing, it uses a static route pointing to a VRRP IP address shared by all routers that could possibly connect SLB-A with the VM subnet.

6. Since the VLANs connected to these routers are extended, they each only have one VRRP IP address. SLB-A uses it to route client connection packets to VM2.

Packets from the client-to-server traffic traverse the DCI network twice. Therefore, they generate a zigzag effect called *tromboning,* which can be disastrous or a minor nuisance, depending on the characteristics of the DCI network.

Corporations with data centers that are not too far apart, which are usually connected via optic fiber, do not mind this effect as much as organizations with DCI networks with lower bandwidth and higher latency. For the latter organizations, tromboning can generate both link saturation and delayed response times for applications.

Going back to Figure 4-12, a much worse traffic issue occurs in the server-to-client traffic, as the following sequence of events details:

1. VM2 sends packets to its default gateway, which is a VRRP IP address that is currently active in site A.

2. Because of the SNAT operation, traffic is directed towards an IP address on SLB-A.

3. A default route on SLB-A points to FW-A. Because client-to-server traffic originally traversed FW-B, FW-A does not recognize the response traffic on its connection table and drops all of its packets.

The effect described is called *traffic asymmetry.* Without much difficulty, it can disrupt applications access on multisite data centers.

Network designers have chosen one of two methods to address asymmetric traffic on active-active scenarios: implement *multisite firewall clusters* or some kind of traffic taming via network configurations. Due to the costs involved with the operationalization of such special firewall clusters, the majority of organizations have opted to deploy network tweaks such as the ones exposed in Figure 4-13.

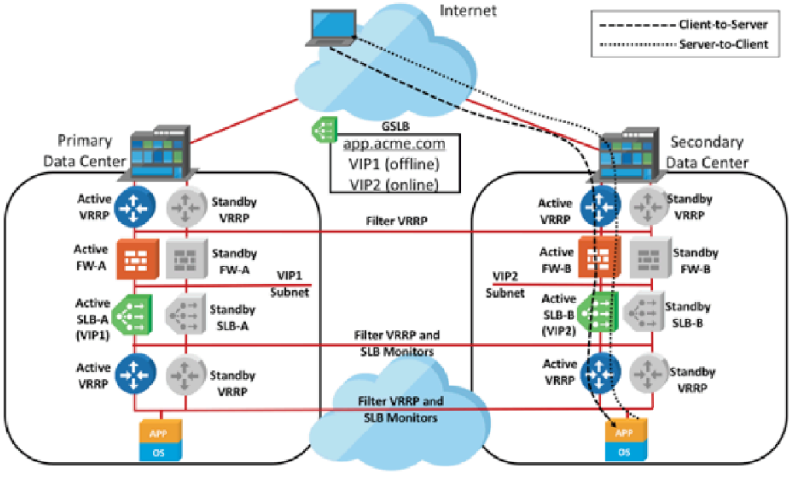


Figure 4-13 Classic Traffic Techniques for Active-Active Sites

Figure 4-13 depicts four different techniques to control client-to-server traffic in active-active scenarios.

- **Different VIP subnets per site:** Not extending the VIP subnet L2 domain permits the VIP1 subnet to be advertised preferably via site A edge routers while the VIP2 subnet is reachable via site B edge routers. In the case of failure on all routers or links from one site, both subnets can be advertised through the alternate site.

- **Global Server Load Balancing (GSLB):** A specialized appliance monitors VIP1 and VIP2 to check whether they are active or not. With such information, the appliance serves as an authoritative DNS server for the app.acme.com domain and load balances clients between both sites according to an algorithm. If one VIP fails, its IP address is not advertised to clients that request the application URL.

- **VRRP Filter on Extended VLANs:** To allow one active VRRP IP address per site and avoid the tromboning effect, VRRP control messages should not be allowed to traverse the DCI network. To effectively avoid devices at one site using a VRRP IP address from the other site, the filter should also block ARP messages.

- **SLB Monitor filter:** SLB monitors are filtered to prevent SLB-A from detecting VMs at site B and vice versa. Each SLB will only balance traffic to VMs that are on its site, with its VIP going offline if all VMs have migrated to the other site.

Note: Please do not consider this best practice list as a mandatory design for extending VLANs. They are purely optional, as data center network architects must decide if their implementation complexity outweighs their benefits.

Provisioning Complexity

Taking into account all challenges and traditional network tweaks explored in the previous two sections, the network provisioning process for a new application capable of running on active-active sites can be particularly convoluted.

To better explain this statement, Figure 4-14 shows all devices that comprise an active-active topology with two data center sites.

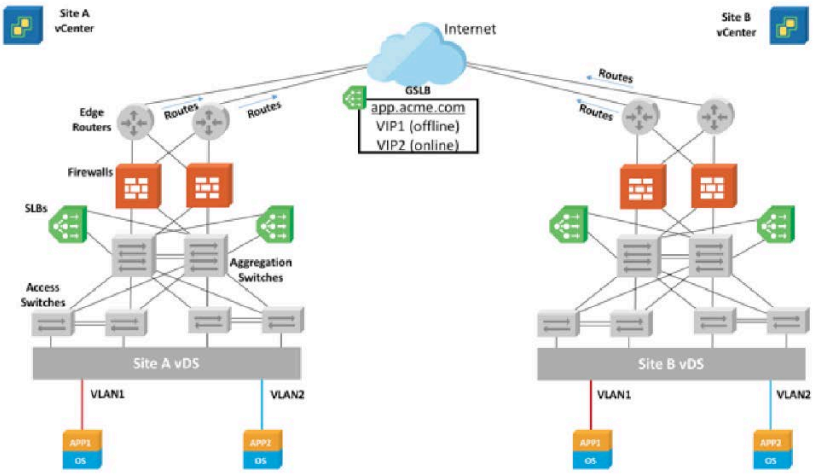


Figure 4-14 Full Active-Active Topology

Glancing at Figure 4-14, for every new application, a network operator should execute the tasks described in Table 4-3.

Table 4-3 Network Operations for New Application on Active-Active Sites

Type of Operation	Configured Devices	Brief Description
Layer 2	**Aggregation and access switches, vDS, Layer 2 extension, firewalls, and SLBs on both sites**	Create VLANs and add them to all trunks. Extend these VLANs with VRRP and SLB monitors filters.
Layer 3	**Firewalls, SLBs, Edge routers on both sites**	Configure IP addresses on firewalls, SLBs, or aggregation switches depending on which device will be the VMs default gateway; configure dynamic route advertisement or static routes on firewall, SLBs, aggregation switches and edge routers.
Security	**Firewalls on both sites**	Create synchronized rules on both sites to protect the new application.
Application Availability	**SLBs on both sites**	Configure server pools with all possible VMs that can be hosted on a single site, monitors, and VIPs which have distinct subnets per site.
Global Server Load Balancing	**GSLB**	Register VIPs to a URL; configure VIP monitors and load balance algorithm.

Hopefully this extensive list of network and security provisioning tasks for a new application communicates the complexities involved in a full active-active topology. As previously stated, complexity means fragility. A seasoned network engineer could make this house of cards stand, but they would surely be required to personally lead all troubleshooting processes in the following years. Documenting and explaining all configuration details to a rookie engineer would be a challenge on its own.

VMware NSX for Active-Active Sites

The full potential of NSX Cross-vCenter is unleashed in active-active data center scenarios. Constructs such as universal logical switches, universal distributed logical routing, universal distributed firewall, and logical load balancing greatly optimize the deployment of new applications at distributed data centers as they were a single location.

Figure 4-15 pictures a VMware Cross-vCenter topology with two sites - Site A and Site B - as a canvas for the discussions that will follow.

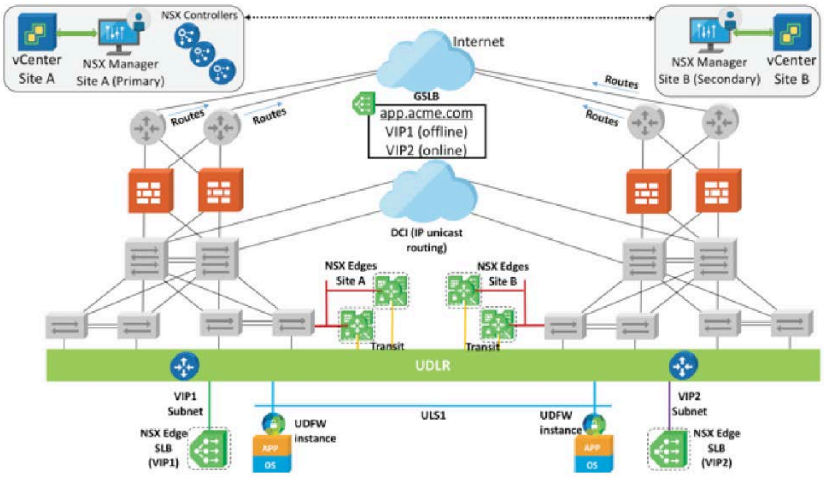


Figure 4-15 VMware NSX Cross-vCenter Topology for Two Active-Active Sites

Working from the foundation discussed in the "VMware NSX for Active-Standby Sites" section, this example delves into the operational tasks that are required to deploy a new application at sites A and B. Table 4-4 outlines the tasks involved.

Table 4-4 VMware NSX Cross-vCenter Operations for New Application Insertion on Active-Active Sites

Type of Operation	Configured Devices	Brief Description
Layer 2	Primary NSX Manager	Create universal logical switches to allow automatic L2 communication between sites via overlay protocol over the DCI network, whose only prerequisites are IP unicast routing and MTU increased according to the used overlay protocol.
Layer 3	Primary NSX Manager	Configure UDLR interfaces for each new ULS. These interfaces will work as a distributed default gateway present on every host at both sites. The NSX Edges will learn these new subnets from the UDLR and redistribute them via BGP to the Edge Routers, with VIP1 subnet being advertised with more precedence via site A and vice versa to normalize ingress traffic.
Security	Primary NSX Manager	Configure UDFW for the new application VMs, which should also include a rule for VMs at site A blocking monitors from the NSX Edge with logical load balancing from site B and vice versa.
Application Availability	Primary and Secondary NSX Managers	Create and configure NSX Edges deploying logical load balancing in one-armed mode connected to new universal logical switches and with active-standby high availability at each site.
Global Server Load Balancing	GSLB	Register VIPs to a URL, configure VIP monitors and load balance algorithm.

Note: Additionally, *egress optimization* can be configured on hosts at each site to guarantee that they will always prefer to send packets toward the site local NSX Edges.

Note: Physical firewalls can be configured with more generic rules (i.e., super-netting all new subnets) for client-to-server traffic to avoid rule configuration as new applications are added.

Note: NSX supports integration with F5 DNS for global server load balancing. For more details, please access: https://blogs.vmware.com/networkvirtualization/2017/05/multi-site-active-active-solutions-nsx-f5-big-ip-dns.html/

When compared to the operational tasks that are required on physical networks to provision networking and security for new applications on active-active sites, the following advantages can be easily observed:

- **Fewer management points**: Primary and secondary NSX Managers and GSLBs are the only points of change. This greatly simplifies such deployments and facilitates network automation, as Chapter 5, "Industrializing Networks" will explain. As many configurations are done only once on the primary NSX Manager, NSX Cross-vCenter scales easily if more sites are added in the future.

- **Layer 2 Extension:** Cross-vCenter does not require the addition of specialized appliances to extend L2 domains to other sites. At its simplest level, this design requires a DCI with IP unicast routing and an increased MTU.

- **Reliability:** NSX deploys *ARP suppression*, which minimizes Address Resolution Protocol messages within local switches. NSX Universal Controllers learn VM IP addresses and enable hosts to locally answer ARP requests, diminishing broadcast traffic that can cause undesired events such as tromboning and loops.

- **Performance:** Universal logical switches, universal distributed logical routers, and universal distributed firewall are executed at the kernel level of the hypervisor, which scales out performance with the addition of new hosts. It is simpler and easier to insert an NSX Edge to perform SLB duties for new applications rather than a physical appliance.

Note: More details about VMware NSX multi-site designs can be found at https://communities.vmware.com/docs/DOC-32552

Layer 2 VPNs

Although NSX Cross-vCenter is well-suited for active-active sites, one of its requirements for the DCI network is sometimes unavailable: the MTU increase required due to the use of overlay protocols.

As an example, some metro-Ethernet service providers cannot make this change on their LAN-to-LAN services due to equipment limitations. In another case, many corporations would like to use the Internet as their DCI network, as it may be the only available IP network connection between a data center and a public cloud.

To fulfill such purposes, VMware has developed *Layer 2 VPN (L2VPN)* as an alternative to L2 extension between an NSX-powered network and other active data center sites. Distinct from Cross-vCenter designs, L2VPN can be used to provide L2 extensions to data centers that are controlled by different administrative entities.

As mentioned in Chapter 2, L2VPN is yet another service deployed by NSX Edges. When deploying such a service, two NSX Edges establish a long-term SSL connection to weld together L2 logical switches and VLANs - in any combination - that belong to distinct sites.

Figure 4-16 shows a relatively simple example of L2VPN between two sites.

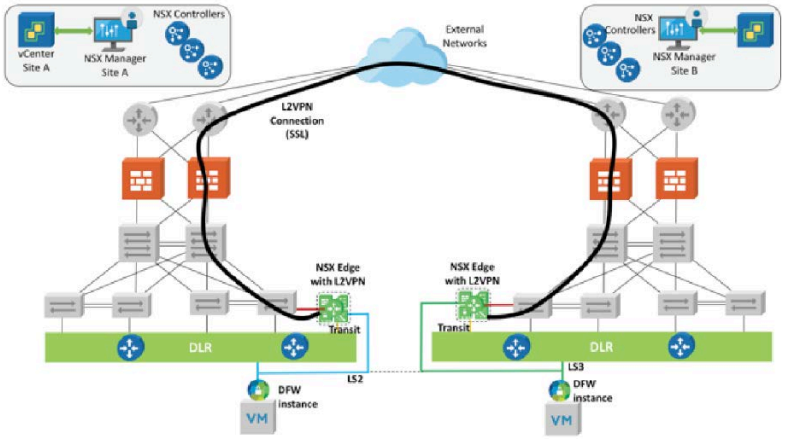


Figure 4-16 L2VPN Example

In this example, the same NSX Edges used for the physical-virtual communication at each site can also deploy L2VPN. With such an arrangement over a standard IP network such as the Internet, this pair of virtual appliances allows logical switch **LS2** at site A to exchange Ethernet frames with logical switch **LS3** at site B as if they were a single broadcast domain.

Although L2VPN does not require any MTU change on the transport network, it is not executed at the kernel level like NSX Cross-vCenter scenarios. Therefore, there is a performance limit measured in gigabits per second for each L2VPN appliance. Always refer to VMware documentation and, when designing this service, take into account the amount of bandwidth available between sites, expected bandwidth between workloads, and other services executed on the same NSX Edge.

Note: Dedicated NSX Edges for L2VPN are considered a best practice for scenarios that require high performance. Nonetheless, when an NSX Edge is deploying both edge routing and L2VPN services, it is possible to configure *local egress optimization* in order to route packets upstream toward the physical network in the same site.

The example depicted on Figure 4-16 does not exhaust all L2VPN scenarios. Another available option would be to license NSX only on one site via *standalone Edges* installed on remote sites.

Essentially an Open Virtual Appliance that can be deployed directly on ESXi hosts or via vCenter, a standalone Edge can be used in different situations as Figure 4-17 explains.

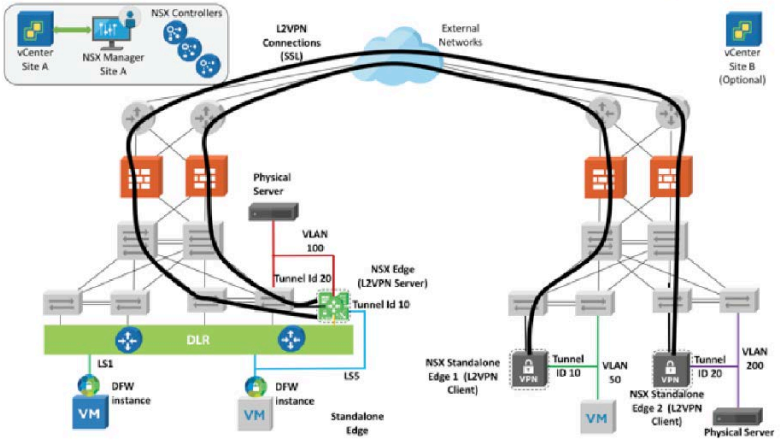


Figure 4-17 L2VPN Design with Standalone Edges

In Figure 4-17, an NSX Edge running at site A is performing the role of a *L2VPN server* connected to two standalone Edges serving as *L2VPN clients*.

Note: VMware NSX supports up to five L2VPN clients per server, and standalone Edges only support L2VPN service. The NSX Standalone Edge only support L2VPN service as a client.

In the L2VPN between the NSX Edge at site A and NSX Standalone Edge 1 at site B, logical switch **LS5** is connected to **VLAN 50** that is connecting a single VM. In the other L2VPN connection between the NSX Edge and NSX Standalone Edge 2, **VLAN 100** from site A is connected to **VLAN 200** at site B, allowing L2 communication between *physical servers* connected to each of these VLANs.

Note: The assignment of L2VPN cross-site pairs of logical switches or VLANs is done via a *tunnel identifier* (Tunnel ID) that is configured on each Edge deploying the L2VPN connection. In Figure 4-17, LS5 and VLAN 50 use Tunnel ID 10 on their connected edges whereas VLANs 100 and 200 use Tunnel ID 20.

VMware NSX L2VPN is great option to allow L2 extension to hybrid cloud implementations. This functionality is already supported on offers such as IBM Cloud and VMware Cloud on Amazon Web Services (VMC on AWS).

What Did This Chapter Achieve?

In summary, this chapter has introduced and explored the following concepts:

- **VMware NSX Cross-vCenter for active-standby sites:** In active-standby scenarios, NSX Cross-vCenter guarantees the synchronization of network and security parameters at both sites via universal objects that are automatically replicated to multiple vCenter domains.

- **VMware NSX Cross-vCenter for active-active sites:** In active-active scenarios, VMware NSX creates a single virtualized data center over multiple distributed sites by actively leveraging the same universal objects across sites. These network and security constructs highly simplify provisioning, L2 extension, and traffic behavior while maintaining each site as a separate failure domain.

- **Layer 2 VPN:** Via NSX Edges and NSX Standalone Edges, this service extends L2 broadcast domains (i.e., logical switches and VLANs) across networks configured with a standard MTU of 1500 bytes, such as the Internet. This service is especially useful for active-active scenarios with an NSX-powered data center connected to service provider sites or public clouds.

Industrializing Networks

> *"The Industrial Revolution has two phases: one material, the other social; one concerning the making of things, the other concerning the making of men."*
> *Charles A. Beard*

All industrial revolutions occurred when the demand for a wide group of assets far surpassed the existent methods of production. Further illustrating these seismic shifts is a concise summary of industrial eras with their respective main achievements and enabling technologies:

- **First Industrial Revolution (18th Century):** Mechanization and steam power.
- **Second Industrial Revolution (19th Century):** Mass production and the assembly line.
- **Third Industrial Revolution (20th Century):** Automation and computers.

Many influential social analysts, economists, and technologist believe the world is in the midst of a fourth industrial revolution. This revolution is essentially creating *cyber physical systems* through a combination of mobility, communication, and above all, applications that can easily leverage these resources in innovative ways.

These modern concepts vigorously challenge the sets of tools, processes, and professionals that are collectively known as information technology. As applications require development at an ever more frantic pace to support new business demands, IT is compelled to evolve accordingly.

IT industrialization requires the embracing of principles from previous revolutions such as standardization and automation. Similarly to those occasions, it also calls for a professional transformation that is currently testing many well-established technical careers. Ultimately, in an assembly line, the slowest part of a process defines the overall speed of the process.

With adequate tools and proper training, IT professionals can coordinate multiple operations that were unimaginable before. Through reusable designs, end-to-end monitoring, and quick process adaptation, IT is achieving both speed *and* quality for the first time.

The previous three chapters described a microcosm of such context, explaining why changes to application architectures defied traditional data center networking technologies. They showed how VMware NSX applied virtualization principles that addressed these difficulties with simplicity and agility. As a natural conclusion, this chapter analyzes the role of full network virtualization in a modern IT environment.

Automating IT

In order to enforce automation principles in IT systems, many data center architects design these environments as *assembly lines*, where compute, storage, networking, and security resources are promptly provisioned for a new application projects via well-coordinated processes.

Figure 5-1 exhibits a simplified architecture of an automated data center.

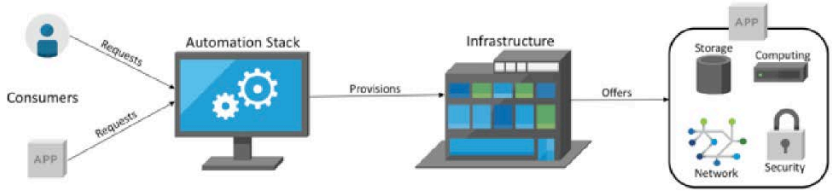


Figure 5-1 Data Center Automation

This figure exposes three basic components of a data center automation architecture:

▪ **Consumers:** Entities that can request IT resources from an automated data center. It may be a human (e.g., end-user, application developer, or IT administrator) or an application.

▪ **Automation Stack:** Translates the consumer requests into infrastructure management operations.

▪ **Infrastructure:** Represents the resources that will be automatically provisioned, including network devices, storage systems, physical servers, virtual machines, security solutions, networking service appliance, and operating systems among others.

The main outcome from such automation process are IT resources that are quickly provisioned and made available for consumption. However, as a first roadblock for their automation endeavor, data center architects usually find an excessive diversity of legacy resources and processes that would neutralize industrialization results such as replicability and predictability.

Therefore, *standardization* is an essential prerequisite for IT automation on all levels. As world-famous industrialist Henry Ford said to his potential car buyers, "*You can have any color you like, as long as it is black*".

Operational teams involved with automated systems must also *develop* procedures that will be executed by specialized automation programs while also rigorously *monitoring* their results. Automation requires the same coding, testing, quality assurance, and versioning as any other software development process.

Within an automation stack there are integrated software modules such as a service catalog, an orchestrator, and a meter, which are respectively developed to offer services to consumers, provision IT resources, and keep track of their usage.

A *service portal* publishes service catalogs, step-by-step wizards for user interaction, fillable forms, approval chains, update messages, usage information, and an *application programming interface (API)* that can expose these same services and information for application consumption.

Whenever a request is issued to the portal, an *orchestrator* interacts with the infrastructure components to execute the changes related to that request. All of these configurations are usually coordinated via a *workflow* that is defined as a sequence of tasks to be executed in order. When the workflow tasks are carried out with success, the orchestrator informs the meter software about the resources changes. This element can then inform the consumer about usage details and billing.

When included in an automation stack, the meter software can be used for *chargeback*, which is a billing process where consumers pay for infrastructure usage, or *showback*, which only presents used resources in order to compare utilization within an organization.

Note: Although many readers may be tempted to consider that the architecture presented on Figure 5-1 fully defines cloud computing, this assumption is not technically correct according to the formal definition of cloud computing published on NIST's Special Publication 800-145. That document lists five essential characteristics of cloud computing - on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service - while the figure, as well as this chapter, primarily focuses on only the first. Nonetheless, when an automation stack provides all essential characteristics of cloud computing, it may be also referred as a *cloud computing stack.*

From an infrastructural standpoint, the automated provisioning of logical resources is much easier than physical systems. Although networking has only a moderate number of logical constructs, it poses specific challenges to data center automation projects. For example:

- Most data center networks are composed of distributed devices whose operation is intrinsically dependent of each other (i.e., *tightly coupled system)*.

- Network configurations are usually executed during maintenance windows to avoid disrupting applications that are in production.

- Network and security services are spread over different types of devices (e.g., switches, routers, firewalls, and SLBs) which, as a general rule, are not centrally managed.

In order to provide a realistic context of data center network automation projects, the next two sections will introduce some sad but important life lessons learned from seasoned network engineers that undertook

such a venture. To draw a fair comparison, the sections will detail traditional network automation approaches that will support the requirements of the same scenario: an automated two-tier application with four VMs, two per tier.

As a user requests this application in the service portal, the following connectivity prerequisites should be satisfied:

- **Front-End Tier:** Both front-end VMs must be in same subnet. Because Internet users will access these machines, they require firewall protection and application availability via server load balancing.

- **Back-End Tier:** Both back-end VMs must also be in the same subnet. Although these servers are not directly accessible to Internet users, they require basic L4 protection from front-end VMs access to avoid lateral attacks. In addition, the back-end service requires application availability via server load balancing.

Furthermore, a service portal user, hereby defined as *tenant*, must only access provisioned resources from other tenants as if they were external users connected to the Internet.

Pre-Provisioned Networks

During the late 2000s, many IT automation projects did not radically change the way data center networks were provisioned. In order to not threaten a production network with potentially dangerous orchestration workflows, more conservative network engineers preferred to prepare a *static network configuration* that would be gradually consumed by applications provisioned via automation stack.

Figure 5-2 illustrates this approach while also shedding light on the details of such implementations.

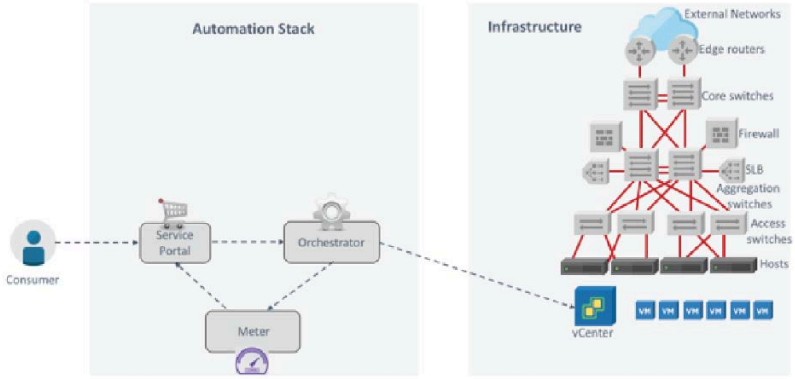


Figure 5-2 Automation with Pre-Provisioned Network

In Figure 5-2, consumer requests are issued to the service portal, which drives the orchestrator to interact with a VM manager - VMware vCenter Server in this example - to create virtual machines. The orchestrator connects them to pre-provisioned VLANs in the physical infrastructure via pre-existent port groups in the virtual switches.

While it would be technically possible to offer to portal users the option to select these port groups to their VMs, this decision could easily compromise the inter-tenant security required on the two-tier application as explained in section “Automating IT”. Consequently, the automation stack must handle the correct assignment of virtual machines to port groups.

Figure 5-3 represents a possible logical topology whose static configuration can support such application.

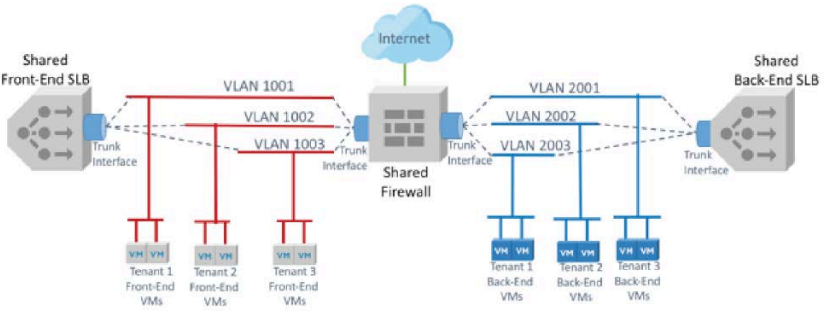


Figure 5-3 Pre-Provisioned Network Topology for Two-Tier Application

In this logical design, each tenant uses two VLANs - front-end VLAN 1XYZ and back-end 2XYZ for tenant XYZ - to host a /29 IP subnet. With the firewall and both SLBs deploying trunk interfaces with IEEE 802.1Q protocol, five IP addresses are used: two for VMs, one for a firewall subinterface serving as default gateway for the VMs on each VLAN, and two for the SLBs - one for the subinterface associated with the and another one for the VIP.

Because a considerable number of tenants are expected in this project, both firewall and SLBs are physical devices that will be shared among all tenants.

Note: Firewall and SLB standby devices are not shown in the figure for a better visualization. Although both shared SLBs could be consolidated into one device, two are represented to also allow a better graphical representation.

The following preconfigured firewall rules must be inserted in the shared physical firewall for Tenant 001 for this example:

- Allow and protect Internet traffic and all other VLANs to front-end VLAN 1001.
- Allow and protect traffic from front-end VMs in VLAN 1001 to back-end VLAN 2001.

Both SLBs are configured in one-armed mode with VIPs created for each VLAN, front-end and back-end, with each pair of VMs as their respective server pool. Although it would be possible to use DHCP to assign IP addresses to the VMs, the orchestrator configures static IP addresses in each VM because they need to be correctly reflected on the SLBs configuration.

Only the firewall should take care of IP routing services, preferably with routing disabled on the SLBs to avoid unintended inter-tenant communication.

Some firewall and SLBs platforms could implement *devices contexts*, which can be defined as virtual firewalls and SLBs with separate routing processes and configurations logically deployed within a physical appliance. Because device contexts allowed a better control of resources and isolation between tenants, they were heavily used in multi-tenant scenarios.

Challenges and Limitations

Considering the logical topology from Figure 5-3, the overall scalability of this automation project in terms of tenants is determined by one of the following two factors: number of device contexts or available VLANs. Because the former limit was around a few hundred contexts, greedier automation projects forfeit this device virtualization technique to support approximately 2000 tenants using 4,000 VLANs from the available range of 1 to 4,094.

Because it relies on static network configuration and firewall rules, this approach strained the introduction of new offers in the service portal. As an exercise, try to picture the network configuration changes that would be needed if the following service were offered for consumption:

- Three-tier application with firewall rules for the communication between each tier and with application availability services on each one.
- Insertion of additional VMs into each network segment.

With such simple proposals, a three-tier tenant would consume three VLANs, changing the VLAN ID distribution per tenant. The easiest method would be dividing the range of VLANs ID between two-tier and three-tier tenants, which would invariably impact the upper limit of supported tenants in the project. Also, to allow more VMs in each subnet, their /29 mask would have to be changed.

These changes should also to be reflected on firewalls and SLBs, which would deploy a different set of rules and load balancing configurations for three-tier application tenants.

As a general rule, any change in the service portal would mean significant changes in static network configuration. And adding hurt to the pain, these adjustments could certainly impact the tenants that were already using the network.

The lack of actual network and security automation prevented the offering of advanced services for tenants such as user-configured firewall rules and content-based decisions.

Automating Physical Networks

To better align service portal offers with networking resources, some organizations decided to embark on the misadventure of automating their physical networks.

In the early 2010s, these projects required the inclusion of network and security configurations into orchestration workflows to allow offering of switches, routers, firewalls, and SLBs as on-demand services to service portal users.

Figure 5-4 unpacks this approach.

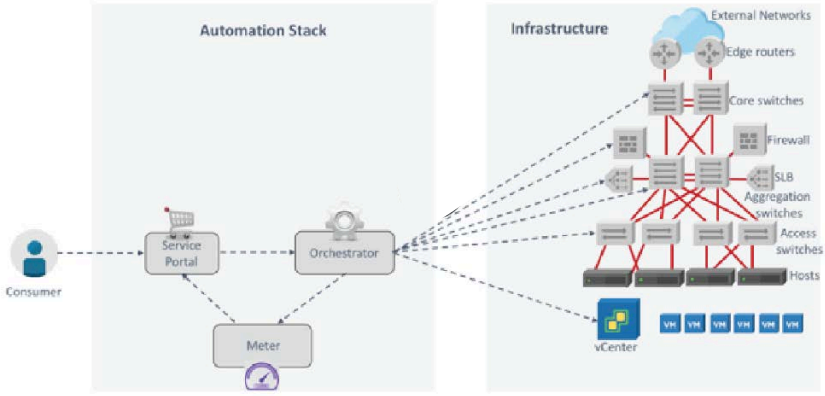


Figure 5-4 Physical Network Automation

Figure 5-4 demonstrates that orchestrators are a critical part of projects that require any kind of networking automation. These software components may use a CLI to communicate with infrastructure components, which through Telnet or SSH sessions can issue commands to and analyze outputs from networking devices according to pre-defined workflows.

Unfortunately, CLIs are hardly standardized as commands and outputs vary wildly among different vendors, platforms, and even firmware versions. Moreover, output analysis in orchestration workflows requires string parsing, a process that can become quite taxing and inexact for workflow developers.

For such reasons, most orchestrators manage devices through APIs, which can be formally defined as a set of functions, variables, and data structures designed to allow software communication. Backing the concept of *web services*, an API regulates how services from a computer system are exposed to the outside world in terms of operations, inputs, outputs, and data format.

When compared to CLI, APIs are considered a powerful alternative for automation because they:

- Can be decoupled from software and hardware implementations
- Do not require parsing since variables are directly exposed via eXtensible Markup Language (XML) data or in any other standardized format
- Usually expose all possible configuration options, different from CLIs and GUIs

Representational State Transfer (REST) is the most used API architecture. Proposed by Roy Thomas Fielding in his 2000 doctoral dissertation "*Architectural Styles and the Design of Network-based Software Architectures*", RESTful APIs are based on web protocols (e.g., HTTP and HTTPS), must be based on client-server requests, must be stateless, and optionally support response caching and responses with coding.

Example of an Automated Network Topology

Using the two-tier application network requirements described in section "Automating IT" as baseline, it is possible to define a typical logical network design that can be incrementally configured as more tenants are added to the data center infrastructure.

Before any automation, all network devices must have their management IP address registered with the orchestrator. During this process, the role of each device must also be declared to allow the orchestrator to identify the operations it should push to each device according to the network topology.

Assuming that the physical network topology is the one depicted in Figure 5-4, Figure 5-5 shows a possible example of a logical network topology that can be used for each tenant.

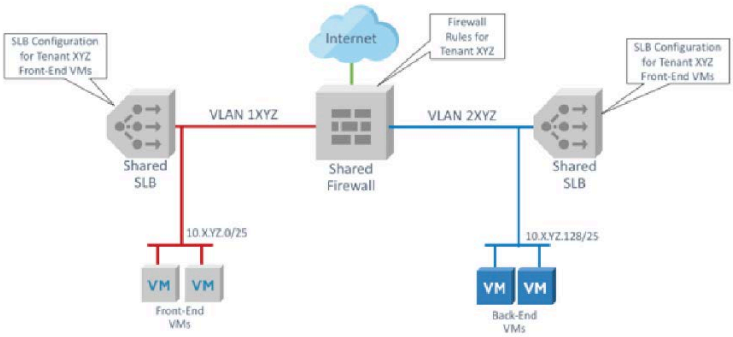


Figure 5-5 Logical Network Topology for Tenant XYZ

As a user issues a request for a two-tier application to the service portal, the orchestrator configures network and security devices to provide all elements exhibited in Figure 5-5.

Table 5-1 sums up possible configuration tasks that are carried out by the orchestrator to support two-tier application tenant on the shared network infrastructure.

Table 5-1 Orchestration Workflow for Physical Network Automation

Type of Operation	Configured Devices	Brief Description
Layer 2	**All aggregation and access switches, firewalls, SLBs, and virtual switches**	Create VLANs 1XYZ and 2XYZ on all switches, adding to each existing trunk including firewalls and SLBs. Create port groups with both VLANs on all virtual switches.
Layer 3	**Firewalls, SLBs, and core switches**	Statically configure the IP addresses on all VMs, 10.X.YZ.0/25 for front-end VMs and 10.X.YZ.128/25 for back-end VMs. Set the IP addresses on firewalls and SLBs interfaces on both VLANs; the firewall will be the default gateway for the VMs. If the shared firewall does not support dynamic routing protocols, both subnets should be statically configured as in core switches to allow reachability from the external networks.
Security	**Firewalls**	Configure firewall rules allowing external traffic, including other tenants, to subnet 10.X.YZ.0/25 and access from this subnet to 10.X.YZ.128/25.
Application Availability	**SLBs**	Configure server pool based on the IP addresses assigned to VMs, monitors to these addresses, load balance algorithm, and VIP on both front-end and back-end VLANs.

Challenges and Limitations

The communication between orchestrator and network devices is the first challenge to be addressed on a physical network automation project. Because many legacy devices do not offer APIs, most orchestration workflows rely on CLI scripts which are fairly complex to develop and maintain. Different types of devices and CLIs can make workflow development even harder, so it was not uncommon that network architects preferred to standardize network platforms and firmware as an attempt to simplify such task, in true "tail wagging the dog" fashion. After all network devices are registered with the orchestration engine, physical topology changes such as adding an access switch would also require additional configuration to the orchestrator.

As an attempt to address both problems, many vendors and open source organizations proposed the use of *network controllers,* which can be defined as network software or appliances that configure other network devices via a *southbound protocol*, while offering a simpler view of the whole network for higher layer software such as an orchestrator via a *northbound* API.

Figure 5-6 portrays the concept of a network controller in more detail.

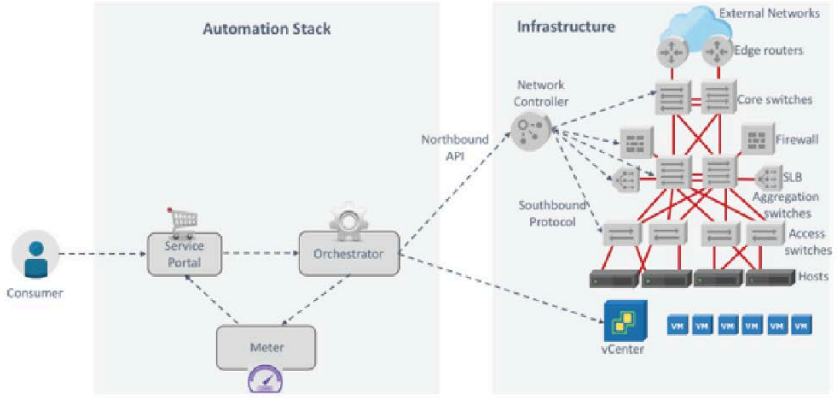


Figure 5-6 Network Automation via Network Controller

Note: Arguably the most famous southbound protocol is *OpenFlow*, which proposes a complete separation of control plane, exclusively executed on its network controller, and data plane in the network devices. As new configuration requests are issued on an OpenFlow controller, it programs flow entries on all managed devices.

While network controllers for physical networks could act as a single point of access to configure network devices, they suffered from shortcomings that prevented a wide adoption. Commercially-available network controllers were usually restricted to a few certified platforms inside each manufacturer portfolio, characterizing a lock-in situation that many organizations would prefer to avoid. These controllers usually did not support firewalls and SLBs, which forced orchestration workflows to directly interact with these devices or their management platform.

Also, *open source network controllers* did not offer the support that was expected on most mission-critical networks, while their implementation and configuration were relatively complex.

With or without controllers, some network engineers preferred to create a new network infrastructure dedicated to automated applications. Although such decision left the non-automated production network untarnished from orchestration-caused disruptions, it created a data center silo that required complex capacity planning and led to low resource utilization.

Network Automation with VMware NSX

Building on the early work done by Nicira with cloud computing providers in the early 2010s, NSX proposed a new approach to network automation without the hurdles and hard choices discussed in the previous two sections.

From an architectural point of view, NSX fundamentally enables simplified integration with automation stacks because it is a centrally managed network and security virtualization platform, as Figure 5-7 demonstrates.

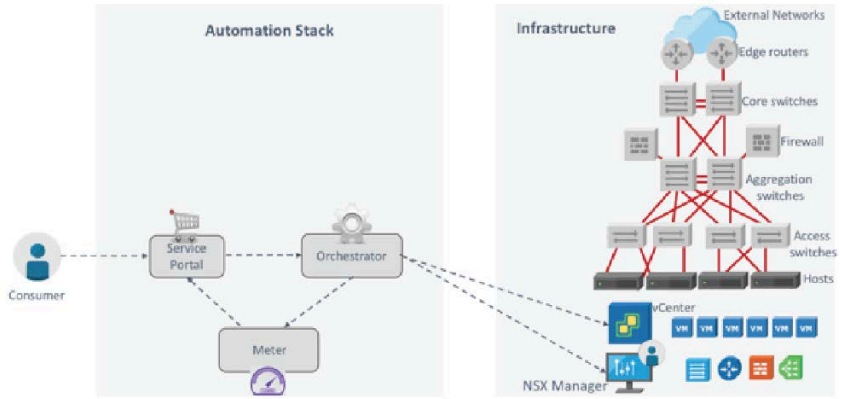


Figure 5-7 Network Automation with VMware NSX

In Figure 5-7, the orchestrator only needs to interact with the VM Manager, represented by VMware vCenter, and an NSX Manager. With access to both virtualization managers, workflows can be created to provision VMs and networks without *any* additional configuration executed on the physical devices, decoupling the provided networking services from the physical network configuration. From a resource utilization perspective, NSX also allows for the production network to be used for both automated and non-automated workloads without the need to create additional network silos.

Note: The full network and security virtualization provided by NSX is a significant step forward from the network appliance virtualization model supported by other network and security vendors. These manufacturers have ported their router, firewall, SLBs software into virtual machines as a way to simplify network service deployments in server virtualization environments. Different from the NSX model, these specialized VMs are licensed per service instance and not per host CPU. Additionally, virtual appliances with distinct services are not centrally managed, which requires individual configuration and monitoring during troubleshooting processes.

NSX Manager also supports a rich and open RESTful API that permits installation, configuration, and monitoring of any of its networking and security kernel-based objects such as logical switches, distributed logical routing, distributed firewall, as well as the many NSX Edge services such as Edge firewall, logical load balancing, network address translation, and virtual private networks.

These constructs and features can be used in varied ways to provide automated virtual networks. As an illustration, Figure 5-8 portrays one possible virtual topology to support the network and security requirements for the two-tier application explained in section “Automating IT”.

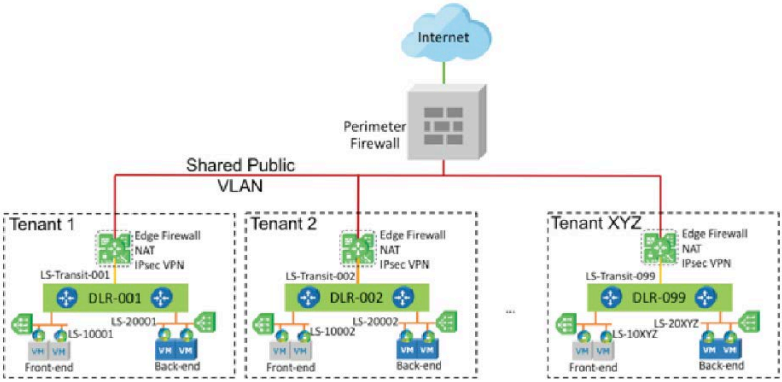


Figure 5-8 Virtual Network and Security Topology

Table 5-2 details all configurations that an orchestrator should likely carry out to deploy such topology as a tenant XYZ is onboarded into the data center infrastructure.

Table 5-2 Orchestration Workflow for Tenant XYZ

Type of Operation	Configured Devices	Brief Description
Layer 2	NSX Manager	Create logical switches **LS-10XYZ**, **LS-20XYZ**, and **LS-Transit-XYZ** - front-end, back-end, and NSX Edge connection, respectively.
Layer 3	NSX Manager	Create DLR-XYZ and statically configure the IP addresses on all VMs, DLR interfaces, internal NSX Edge interface, and SLBs interfaces and VIPs on all tenants. Each NSX Edge uplink interface will have a different IP address, which will be used for user access to the front-end SLB VIP via DNAT. The default routes on the tenant NSX Edges must point to the perimeter firewall interface on the shared VLAN.
Security	NSX Manager	Configure Edge firewall and distributed firewall rules allowing external traffic - Internet and all other tenants - to the front-end subnet. Configure distributed firewall rule allowing traffic from this subnet to the back-end subnet. Additionally, configure IPsec VPN or SSL VPN-Plus to allow secure management traffic to both front-end and back-end VMs.
Application Availability	NSX Manager	Instantiate NSX Edges and enable logical load balancing in one-armed mode on **LS-10XYZ** and **LS-20XYZ**. Configure server pools based on the IP addresses assigned to VMs, monitors to these addresses, load balancing algorithm, and VIP on both NSX Edges. Use X-Forward-For to log client IP address on web access to front-end VMs.

Note: Because all L3 elements are unique per tenant, it is possible to use overlapping subnets to reuse the same four addresses for the VMs in all tenants. If this option is chosen, the firewall rules can be simplified by the use of these shared parameters to avoid customization per tenant.

Besides orchestration simplicity via NSX API, NSX brings the following advantages over physical network automation approaches:

- Logical switches extend the number of available broadcast domains well above the VLAN ID limit of 4,094. Although theoretically it can achieve more than 16 million logical switches, NSX supports 10,000 logical switches per NSX Manager. Such characteristic facilitates the inclusion of new application tiers into a tenant virtual topology.

- The distributed logical router scales performance for east-west traffic between front-end and back-end VMs. As there are multiple DLR instances, one for each tenant, the same subnets and IP addresses can be reused on all tenant elements, including virtual machines, DLR interfaces, and NSX Edges - except for the uplink interfaces connected to the shared VLAN.

- The distributed firewall can enforce security over intra-segment, intra-tenant, and inter-tenant traffic.

- Both NSX Edges deploying SLBs are centrally managed by NSX Manager. They are not licensed by instance, which significantly decreases the overall cost of the on-demand infrastructure.

- The shared firewall rules can be applied to the IP subnet hosted on the shared VLAN. Inter-tenant security can be enhanced by security partner integration applied to the front-end logical switches via service composer.

The network design shown in Figure 5-8 is an example of how NSX can deploy automated virtual networks for an application. Other NSX designs can be applied to the same scenario depending on specific project requirements such as desired scalability of tenants, performance, and security.

The previous example is certainly not the only network automation option with NSX. As an illustration, Figure 5-9 exhibits yet another possible NSX design for these tenants.

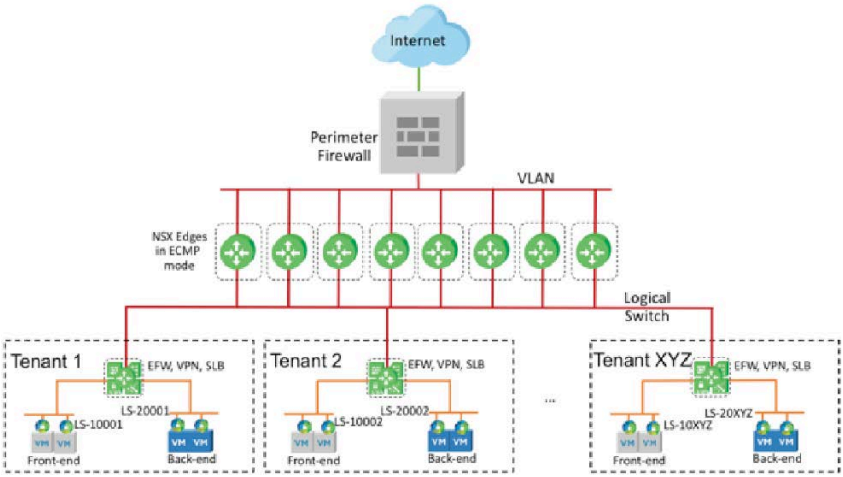


Figure 5-9 Another VMware NSX Tenant Network Design

With the virtual network topology depicted in Figure 5-9, the tenants do not use distributed logical routing because their east-west traffic is not expected to be intense.

Chapter 2, "Rethinking Network Constructs", introduced the concept of an ECMP NSX Edge cluster to scale edge routing for NSX designs. The design depicted in Figure 5-9 allows the deployment of distinct subnets per tenant logical switch. These can be statically configured on the NSX Edges from the ECMP cluster and publicly advertised via BGP to the perimeter or other router. Additionally, the default routes on the tenant NSX Edges must point to the internal interfaces of each member of the ECMP cluster.

Both tenant designs, along with many others, can be used simultaneously as more application offers are added to the service portal. Flexibility is one of the most valued advantages of the full network virtualization in NSX for automated environments.

VMware NSX Integration with Automation Stacks

As mentioned in "Network Automation with VMware NSX", the completeness of the NSX API allows easy integration with potentially any automation stack with a minimally flexible orchestrator.

During its short lifespan, NSX has already achieved tight integration with automation and cloud computing platforms such as VMware vRealize Automation, OpenStack (including VMware Integrated OpenStack), VMware Cloud Director, and VMware vRealize Orchestrator. NSX networking and security services can also be automated via tools such as Chef, Puppet, and Ansible.

Although the description of each aforementioned integration is out of the scope of this book, the next sections will highlight two of the most common automation integrations for NSX.

vRealize Automation

vRealize Automation, or simply vRA, is one of the most complete automation stacks within VMware's portfolio. vRA allows the design of customized secure portals where authorized administrators, developers, and business users can request new services, while ensuring compliance with business policies and management objectives.

vRA is composed of the following components:

- **vRealize Automation appliance:** Executes the web console, service portal, and application services.
- **vRealize Automation Infrastructure as a Service (IaaS):** Provisions IaaS resources in heterogeneous hypervisors, cloud environments, and physical hosts.
- **Authentication Services:** Provides authentication, authorization, and accounting services for administrators and users.
- **vRealize Orchestrator:** Extends vRA to support any data center automation.
- **vRealize Business:** Calculates total costs of a data center, coordinates and compares service prices, and generates reports about expenditures on provisioned resources.
- **vRealize Code Stream:** Automation tool that allows DevOps teams to build a release pipeline and provision new applications in a modern development environment.

The most powerful vRA construct is a *service blueprint*, which comprehensively specifies a service that will be offered on a user portal. Rather than programming complex workflows, blueprints can easily be created through drag-and-drop objects in a design canvas, which is illustrated in Figure 5-10.

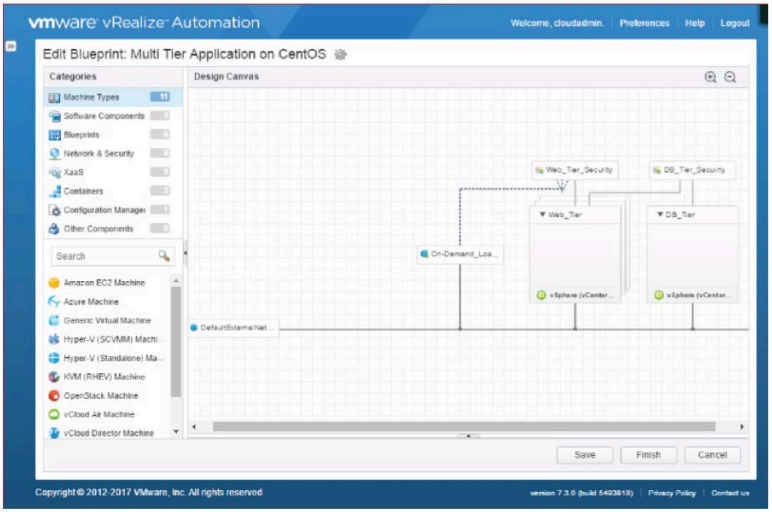


Figure 5-10 vRealize Automation Blueprint Example

As Figure 5-10 shows, blueprints can easily use network and security definitions such as “External Networks”, “On-Demand_Load_Balancing”, and “Security Groups” to deliver network services (e.g., VLAN or logical switches, NSX Edge logical load balancing configuration, and security groups from Service Composer) in order to provide secure connectivity for automated applications.

Besides automating virtual networks via integration with VMware NSX, vRA can also assign virtual machine network parameters such as IP address, netmask, default gateway, DNS server, and Active Directory domain among others. With that intention, vRA keeps a record of assigned IP addresses from a pool and may use the following network profiles to build flexible application network topologies that may need to be created when provision VMs:

- **External:** Used when VMs will be connected to an existing network (i.e., VLAN or logical switch).
- **NAT:** Used when there are overlapping IP addresses across different blueprint deployments and external connectivity to the automated VMs is necessary.
- **Routed:** Used when end-to-end routable access is required to the automated VMs.

Note: More details on VMware vRealize Automation is available at https://www.vmware.com/products/vrealize-automation.html

OpenStack

In 2010, the U.S. National Aeronautics and Space Administration (NASA) along with cloud provider Rackspace created an open source initiative called OpenStack, with the objective to build public and private cloud computing environments.

An OpenStack deployment depends on the operationalization of services that are developed as projects within the OpenStack community. Table 5-3 explains the objectives of the main OpenStack projects.

Table 5-3 OpenStack Projects

Service	Description
Keystone	Provides identity services for all services in an OpenStack installation
Nova	Provisions VMs on behalf of cloud end users
Glance	Coordinates software images that will be used to boot virtual and physical instances within the OpenStack cloud
Neutron	Controls networking resources to support cloud service requests
Swift	Offers data storage services in the form of objects
Cinder	Provisions block-based storage for compute instances
Heat	Manages the entire orchestration of infrastructure and application within OpenStack clouds

All OpenStack modules described in Table 5-3 interact with each other via vendor-neutral APIs. Furthermore, these services can integrate with infrastructure elements (e.g., servers, network, storage devices) through plugins that standardize the operations that will be deployed on these resources.

VMware NSX tightly integrates with a wide range of OpenStack deployments via a plugin for the Neutron module. This module enables API calls from Neutron to NSX Manager in order to create the following correspondence between Neutron and VMware network constructs:

- **Networks:** Logical switches
- **Routers:** NSX Edges (SNAT)
- **Floating IPs:** NSX Edges (DNAT)
- **DHCP services:** NSX Edge (DHCP Relay)
- **Security groups:** NSX security groups
- **Load balancing as-a-service (LBaaS):** NSX Edge (logical load balancing).

VMware Integrated OpenStack (VIO) is a VMware OpenStack distribution that was especially designed for easy installation and scalable use in cloud computing deployments. Distinctive from other OpenStack distributions, VIO does not depend on complex integration with hardware-based third-party devices because it is pre-integrated with VMware vCenter Server and VMware NSX, as presented Figure 5-11.

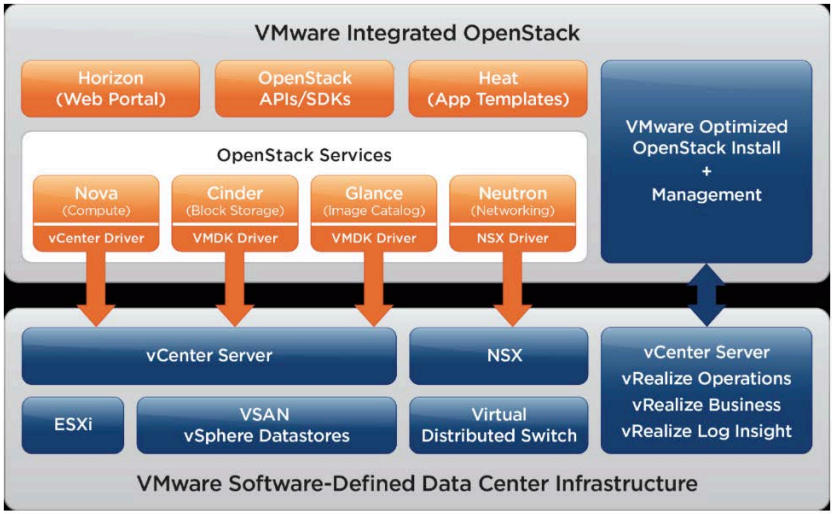


Figure 5-11 VMware Integrated OpenStack Architecture

Figure 5-11 highlights the integration between VIO and VMware Software Defined Data Center (SDDC) components such as vCenter, NSX, and vSAN. This integration greatly simplifies OpenStack installation, upgrade, and troubleshooting through familiar VMware management tools.

Note: More details on VMware Integrated OpenStack by can be found by visiting https://www.vmware.com/products/openstack.html

What Did This Chapter Achieve?

This chapter introduced and explored the following concepts:

- **Challenges of automating IT with a pre-provisioned static network configuration:** Although this method is less complex from a strict networking and security perspective, it invariably paralyzes the service offer in the portal until a new configuration is performed in the network infrastructure.

- **Challenges of physical network automation:** Many network platforms do not offer application programming interfaces, which consequently forces orchestrators to use complex scripting based on command-line interfaces. Additionally, the variety of networking and security devices within a physical data center network infrastructure demand that after any new change in the application topology, a change in the orchestration workflows must be done. Although network controllers initially proposed a solution to this problem, their implementations were limited to a few network and security platforms due to vendor-required support.

- **Advantages of network automation via VMware NSX:** Being a software-based network and security virtualization platform, NSX allows for network automation to be delivered with flexibility, without physical configuration change required, and through a centralized API.

- **VMware NSX Integration:** NSX integrates with VMware vRealize Automation, OpenStack, VMware Integrated OpenStack, VMware vCloud Director®, VMware vRealize Orchestrator, Chef, Puppet, and Ansible among other platforms and tools.

One Network

"Virtual reality is a technology that can bring people places they might not otherwise reach"
John Patrick Pullen

As seasoned CIOs and infrastructure managers can attest, data centers cannot achieve effectiveness only through the acquisition of best-of-breed products. These complex IT environments require such intense integration that unilateral decisions may easily compromise their capacity and evolution.

Data center personnel are commonly organized in technology areas such as applications, servers, storage, facilities, security, and networking. Although these teams may work independently, their individual projects are highly influenced by other areas. Data center networks have always aimed to provide connectivity for hosted servers; however, the rapid evolution of server virtualization imposed serious changes to the way these structures were designed over the last two decades.

This chapter discusses the challenges and technologies that influenced data center architects to carry out the evolution of data center networks in the environments under their responsibility. It also explains how network virtualization has helped thousands of organizations to build future-proof data center networks that combine both physical and virtual elements, while avoiding potential pitfalls such as vendor lock-in.

Finally, the chapter addresses the subject of network operations through full traffic visibility and intelligent log analytics over a consolidated (i.e., physical plus virtual) data center network.

Data Center Network Evolution

A crucial part of any data center project, a data center network exists to transport server data toward clients or between servers. In order to meet the rising expectations regarding modern applications deployed in a data center, its network must embody attributes such as:

- **Availability:** Be resilient enough to react in a timely manner to failures and protect application performance from such connectivity hindrances.
- **Agility:** Easy and fast provisioning of network constructs such as broadcast domains, IP subnets, and security rules among others.
- **Adaptability:** Support future applications, scenarios, scale, and technologies that were not predicted at the time of its original project.
- **Accuracy:** Provide reliable data transport for all server traffic as well as descriptive statistics, events, and error logging.

There are multiple ways of building a network with high performance switches as well as achieving all four aforementioned "A's". The large majority of data centers have deployed at least one of the designs that will be discussed in the next three sections.

Some of these designs were already presented in previous chapters. The next sections will specifically focus on providing a comprehensive understanding of such architecture progression.

Three-Tier Design

The Internet boom in the 1990s motivated many organizations to deploy data centers with the intention of sharing content and applications to external and internal users. As the number of hosted servers escalated, data center networks evolved from a pair of *server farm* switches to a three-tier switch topology, represented in Figure 6-1.

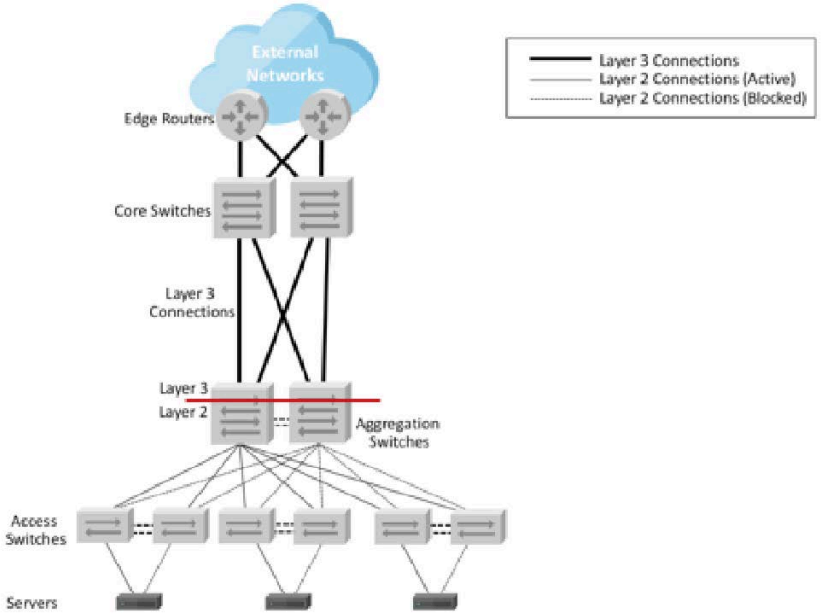


Figure 6-1 Three-Tier Data Center Network Design

As more experienced network engineers may acknowledge, the *three-tier design* closely mimics the traditional three-tier campus design, whose main focus is providing L2 and L3 connectivity for desktop users.

Similarly, in a three-tier data center network design, each switch tier serves specific roles. The *access tier* takes care of server connections and usually contains the largest number of Ethernet ports in the topology. Data center network architects commonly deploy simple L2 switches in this tier, in *end-of-row (EoR) or top-of-rack (ToR)* physical topologies.

The *aggregation tier* provides a confluence point for server L2 domains (i.e., VLANs) as well as an ideal location for networking service insertion such as default gateways, firewalls, and load balancers.

Finally, the *core tier* exists to connect aggregation pairs and external networks (e.g., the Internet or the organization campus network). Architecturally, core switches provide a way of scaling a data center network through the addition of more aggregation pairs.

Note: Many small and medium data centers, which may not require an extensive number of access ports, have chosen to collapse tiers to avoid unnecessary hardware acquisition.

Even though three-tier campus and data center networks share the same kind of network devices (i.e., Ethernet switches), their topologies present fundamental differences:

- **Distribution vs aggregation switches:** Campus networks use distribution switches to concentrate access switches of the same building. Aggregation switches are used to connect servers that require L2 connectivity among them or with networking services such as server load balancers and firewalls. These are not represented in the Figure 6-1 for the sake of clear visualization.

- **Number of impacted users:** A disrupted Ethernet switch in a campus network can affect a group of users. A failed data center switch can potentially interrupt application access to millions of clients.

- **Desktop vs server connections:** A user computer usually requires one Ethernet connection to a campus access switch, while a server must deploy redundant connections to alleviate the effect of a link failure.

- **Traffic interest:** Desktops and mobile devices in a campus network are usually interested in applications that are located in external networks such as the Internet or a local data center. As discussed in Chapter 2, "Rethinking Network Constructs", data center networks commonly transport more east-west (i.e., server-server) than north-south (i.e., client-server).

As the aforementioned chapter described, L3 connections can easily leverage ECMP and dynamic routing protocols to provide simple traffic distribution and failure rebound. For L2 connections, all three-tier data center networks still rely on STP and its variations to avoid Ethernet loops and mitigate switch and connection failures.

As an expected collateral effect, three-tier designs introduced a high number of blocked links, wasting approximately half of deployed network resources such as switch ports, optical adapters, and cabling. For the sake of simplicity, server connections deployed active-standby links to access switches in order to avoid active-active mechanisms that were specific to some network interface controllers or operating systems.

Moreover, uplink failures could require STP to converge again, disrupting all server connectivity in a L2 domain during a few precious seconds, which was unacceptable for many real-time business applications.

Three-Tier Design with Multi-Chassis Link Aggregation

In the mid-2000s, several data center architectural changes started pushing the limits of the three-tier design. As in Chapter 3, "Virtualization as a Security Control", applications started to be designed in multiple component layers while server virtualization was gaining momentum in most enterprise and service provider data centers.

In an attempt to also optimize cabling and topology investment, network vendors proposed multiple different implementations of what can be defined as multi-chassis link aggregation. This virtualization technique is based on two switches behaving as a single entity. This is done to fool the Spanning Tree Protocol and allow two or more active uplinks for switches deploying standard link aggregation protocols such as IEEE 802.3ad. Although STP does not run between access switches and physical servers, multi-chassis link aggregation also allows all server connections to be active via IEEE 802.3ad.

Figure 6-2 represents the direct outcome of multi-chassis link aggregation.

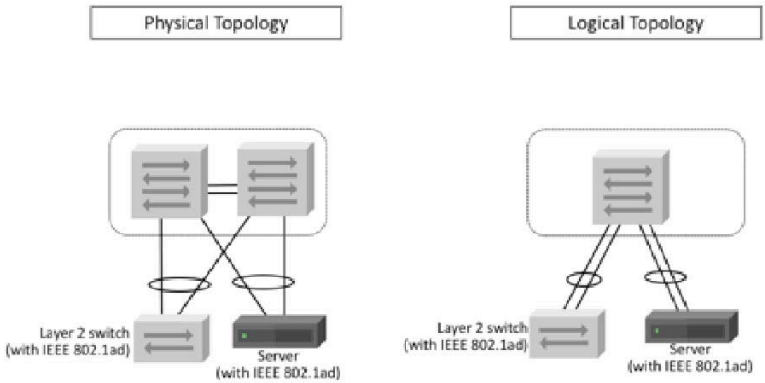


Figure 6-2 Multi-Chassis Link Aggregation

Figure 6-2 presents two different devices deploying standard link aggregation toward a pair of upstream switches in multi-chassis mode: an L2 switch and a server. From a logical perspective, the upstream switches are actually a single switch for both devices.

Specifically for the switch, the spanning tree instance formed does not contain three switches but only two; a great simplification that cleverly avoids blocked links.

With many networking vendors creating their own unique implementation of multi-chassis link aggregation, data center network architects developed a slight variation of the three-tier network design, which is diagrammed in Figure 6-3.

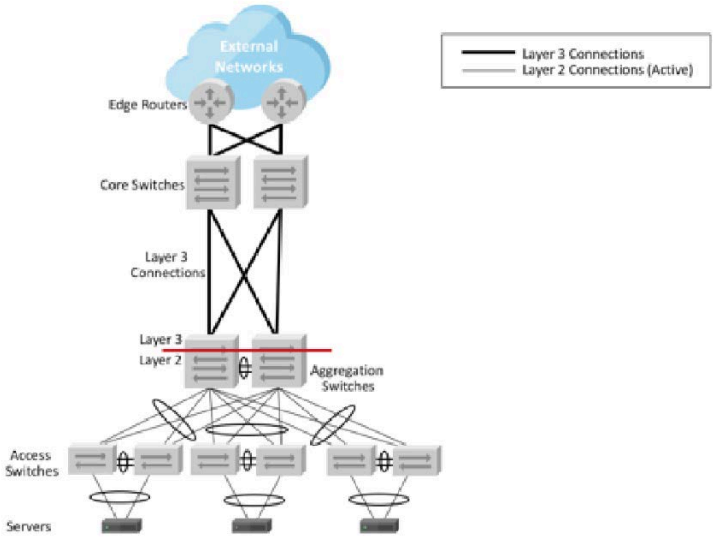


Figure 6-3 Three-Tier Design with Multi-Chassis Link Aggregation

In Figure 6-3, both aggregation and access switches are deploying multi-chassis link aggregation, which allows all L2 links to actively transport traffic.

This design doubles the available bandwidth when compared with the STP-based three-tier topology. The simplification of spanning tree instances increases the overall stability of the data center network. Now, a failure of an uplink within a link aggregation link can be recovered in milliseconds, a great improvement over standard STP convergence.

Such outcomes are interesting considering that server virtualization was actually increasing the traffic each host sends to and receives from the network. However, larger data centers started observing that the number of ports used at each aggregation switch depended on two factors: the number of required access switches and the number of available uplinks to each access switch. As more virtual and physical servers were added to such topologies, the aggregation switches became a scalability bottleneck.

While most networking vendors offered extremely dense switches, some models with more than 700 ports, many network designers considered such devices too expensive and risky to the overall network; a failed switch could impact too many connected devices. These humongous devices only brought short relief to highly-scaled network infrastructure, until their port limit was eventually reached.

For these expanded scenarios, network designers continued to add more aggregation pairs, as Figure 6-4 demonstrates.

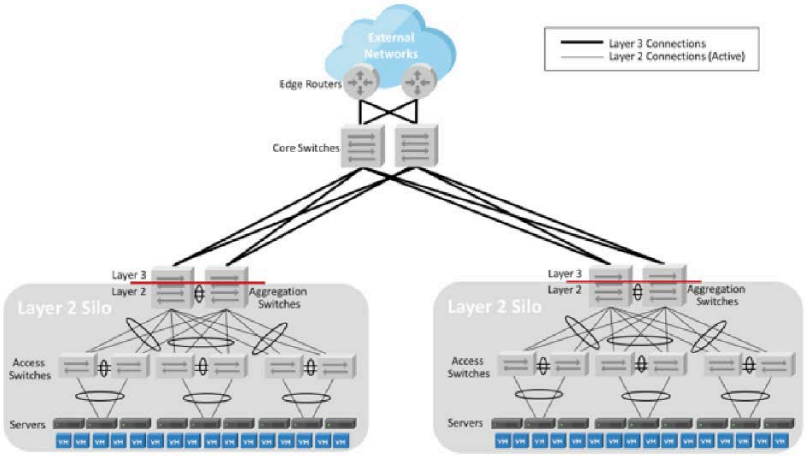


Figure 6-4 Three-Tier Multi-Chassis Link Aggregation Design with Two Aggregation Pairs

In Figure 6-4, another pair of aggregation switches was added to the network topology to accommodate more servers. A side effect is introduced to this data center network: there are now two L2 silos that cannot share broadcast domains - VLANs in the described scenario - due to the L3 network that separates them.

These silos bring complexity to server operation because they forbid the provisioning of a VLAN to all the racks in this facility. This locks servers - virtual or physical - that require L2 connectivity to the same silo/ quadrant/block/pod/cell, depending on how each organization names these network substructures.

The described situation continues to challenge many data centers, forcing data center architects to deal with server capacity planning inside each silo. For example, if computing resources were exhausted in Silo 1, more servers would have to be added to this domain because free resources in Silo 2 cannot be deployed to the same VLANs from Silo 1.

Although many service providers solved this problem by reallocating the L2/L3 border to the core switches, most network designers dread this alternative for one reason: all aggregation and access switches would be under the same failure domain and subjected to extensive STP convergence time.

Until the next significant network architecture evolution, data center architects preferred to deal with L2 silos associated with line-of-business and security policies of an organization.

Ethernet Fabrics

In an attempt to address the architectural challenges of the three-tier design described in the last section, many vendors started to offer data center network designs called *Ethernet fabrics*.

In the context of networking, the term *fabric* comes from storage area networks (SANs) based on Fibre Channel protocol. This protocol recommended that a network composed of potentially hundreds of devices should be configured as a single entity. Repackaging the term to data center networks and leveraging L2 multipathing protocols such as TRILL (Transparent Interconnection of Lot of Links) or proprietary implementations, these manufacturers recommend the use of Ethernet fabrics in spine-leaf topologies as a scalable solution for data centers.

Figure 6-5 depicts one example of an Ethernet fabric using such a topology.

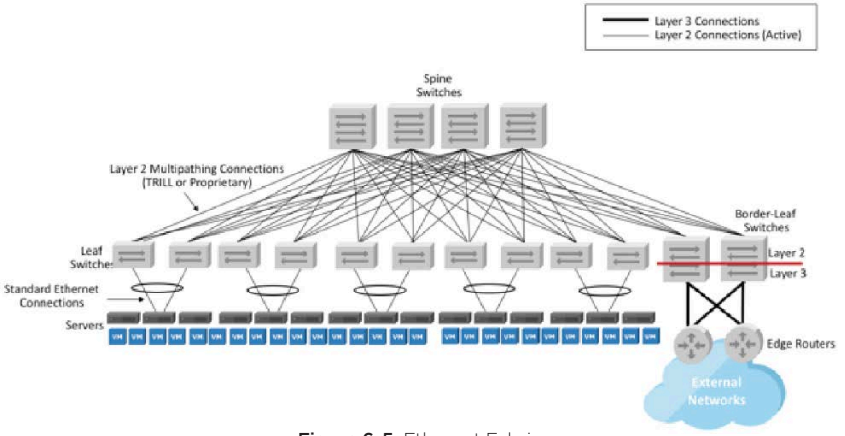


Figure 6-5 Ethernet Fabric

In Figure 6-5, an L2 multipathing technology (e.g., TRILL, Cisco FabricPath, or Brocade Virtual Cluster System among others) enables all L2 connections between compatible switches to carry traffic. Because more than two L2 paths can be active, switches can be organized in a spine-leaf topology.

Servers are connected to leaf switches via standard Ethernet connections, which can sometimes leverage multi-chassis link aggregation, whereas specialized border-leaf switches connect the fabric to external networks such as the Internet or internal campus.

Through spine-leaf topologies, Ethernet fabrics can respond to an increasing number of ports via the insertion of more leaf switches. If more bandwidth is required between leaf switches, this topology allows the addition of more spine switches, while maintaining a single L2 domain shared by all switches.

Although this strategy is attractive for many network architects, the adoption of Ethernet fabrics is still not as widespread as their manufacturers expected. The reasons for this slow migration from three-tier topologies include:

- Most Ethernet fabric implementations require switches from the same network vendor, sometimes even the same platform inside a vendor's portfolio. Many data center architects feel uncomfortable with such vendor lock-in, which can bring risk and inflexibility to the future of their environments. Within each implementation, each vendor included differentiators such as network controllers which further decreased the solutions openness.

- Different from a migration from three-tier topologies to multi-channel link aggregation topologies, the evolution to Ethernet fabrics generally requires a rip-and-replacement of network switches along with considerable cabling adaptation.

- Network services such as firewall and load balancers are usually concentrated on a special pair of leaf switches called service leaves. If these services are used for east-west traffic, these switches may be overloaded with hair-pinned traffic. Such traffic is not represented in the Figure 6-1 for the sake of clear visualization.

- Depending on the proprietary features implemented such as distributed default gateways among the leaves and stateless security rules, these fabrics may not scale as expected.

Data Center Consolidation through Network Virtualization

In 2013, VMware NSX was released in the market as the first enterprise-class network virtualization platform. As the section "Ethernet Fabrics" described, architects from large data centers were hesitating between two possible evolution scenarios during this period:

1. Continue to deal with network silos and manually control the distribution of resources (e.g., computing, storage, and applications) between them

2. Migrate the existent topology to a spine-leaf Ethernet fabric

VMware NSX requires only IP unicast routing and an MTU adjustment in the physical network to deploy all of its network virtualization features. Considering that all three data center network designs discussed in the previous sections can provide such prerequisites without much difficulty, it is valid to conclude that the solution can also help data center architects to overcome network challenges on three-tier topologies without hardware rip-and-replace.

Figure 6-6 clarifies this statement.

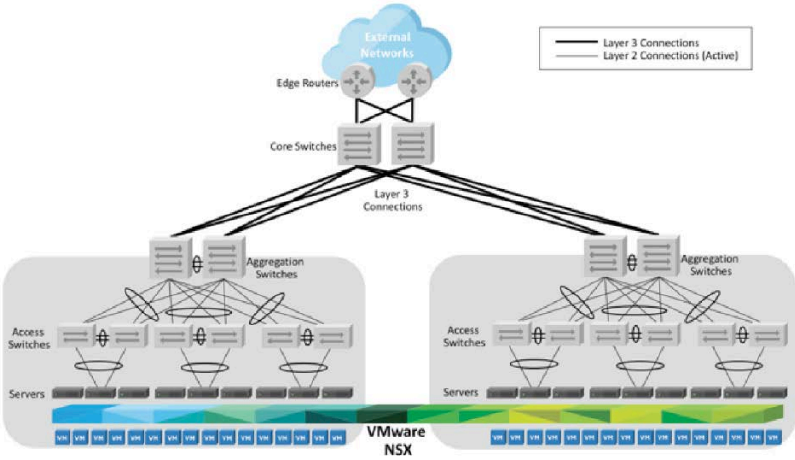


Figure 6-6 Data Center Network Consolidation with VMware NSX

This example suggests that NSX can connect applications across multiple L2 silos to provide compute resource optimization as well as VM mobility across all data centers.

Figure 6-7 delves deeper into an NSX topology, providing data center consolidation for a three-tier design with two L2 silos.

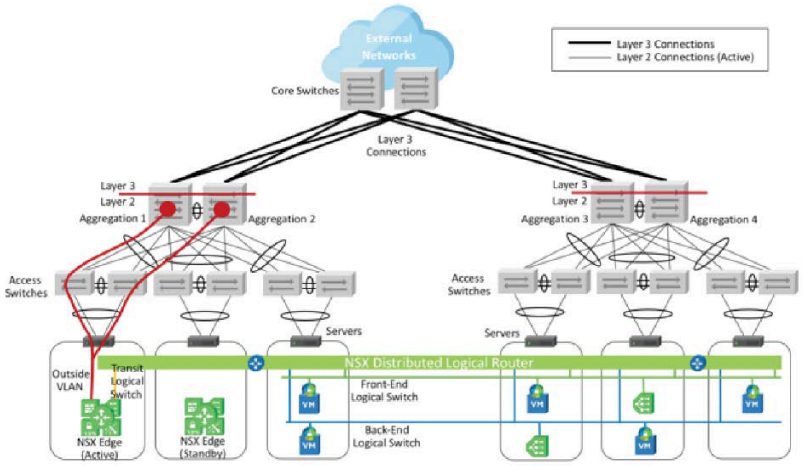


Figure 6-7 Data Center Consolidation

In Figure 6-7, an application composed of front-end and back-end virtual machines uses compute resources from both network silos through the use of logical switches. In this scenario, east-west traffic is optimized by distributed logical routing and adequately protected by distributed firewall rules. Logical load balancing is also deployed in one-armed mode at each logical switch with total mobility of virtual machines between both network silos.

Because the NSX Edge is a VM running on a host that usually requires a transit logical switch and an outside VLAN connection to the physical network, it could potentially be located at any host. However, this design would also require the provisioning of this VLAN on every port for every host, which is possible for access switches connected to aggregation switches 1 and 2, but not for switches connected to aggregation switches 3 and 4.

To facilitate the provisioning of these special VLANs, scalable VMware designs recommend dedicated hosts for NSX Edge provisioning, called *Edge clusters*. Management virtual machines such as VMware vCenter Server, NSX Manager, and NSX Controllers are usually deployed on another separate set of hosts called the *management cluster*.

Note: Some medium-size VMware NSX designs consolidate the management cluster with the Edge cluster, while small NSX implementations can consolidate all three functions (i.e., management, edge, and compute) into a single consolidated cluster. More details about VMware NSX sizing can be found on https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/nsx/vmw-nsx-network-virtualization-design-guide.pdf

An Open Data Center Network Architecture

An NSX deployment can ease network and security provisioning over any data center network design, including both three-tier topologies and spine-leaf Ethernet fabrics. However, data center architects that are already used to the benefits of network virtualization know the platform can also influence the way data center physical networks can evolve.

Many engineers have used NSX to materialize a network architecture that was only available for massive scale data centers from major cloud and application providers: IP fabrics.

Figure 6-8 presents the use of IP fabrics as underlay for NSX.

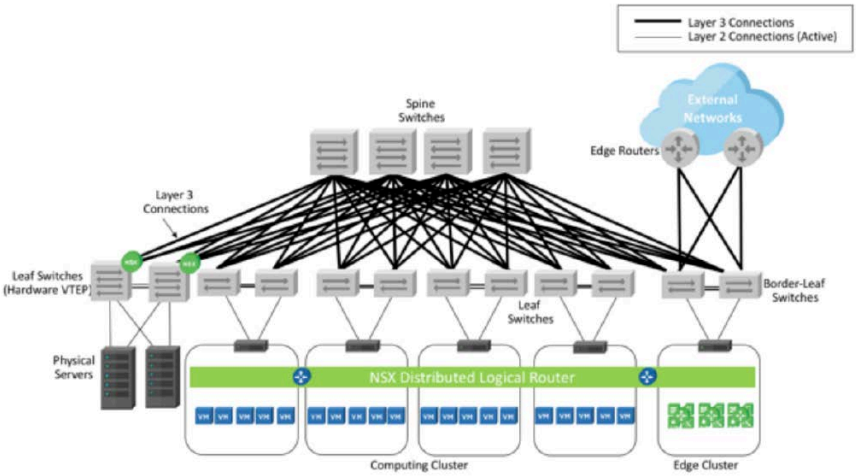


Figure 6-8 Layer 3 Fabric with VMware NSX

In Figure 6-8, the physical network is composed of high performance switches arranged in a spine-leaf topology, linked via L3 connections. It uses standard dynamic routing protocols (e.g., OSPF and BGP) to guarantee IP connectivity between any pair of leaf switches.

Similar to Ethernet fabrics, the spine-leaf topology on IP fabrics can deploy more spine switches for more bandwidth and more leaf switches if more access ports are needed for server connection. However, because of the simple requirements from NSX, an IP fabric is not restricted to a network vendor or specific platform.

With an open underlay, NSX can provide layer 2 and 3 connectivity, security, and logical load balancing for all VMs installed on any host of this data center while also supporting data center mobility.

Furthermore, selected switch platforms from most network vendors are NSX-compatible hardware VXLAN Tunnel EndPoints (VTEPs) - physical networking devices such as switches and routers that can encapsulate and decapsulate Ethernet frames into VXLAN packets to be exchanged with hosts running NSX. These devices allow hardware-assisted L2 communication between physical servers attached to a VLAN and VMs connected to logical switches.

Note: An updated list of hardware VTEPs compatible with NSX can be found at https://www.vmware.com/resources/compatibility/search.php?deviceCategory=hvxg

As traditional network vendors have proposed new data center network architectures which offered more scale and reliability, they also increased hardware dependency at each evolutionary step.

Figure 6-9 summarizes this progression.

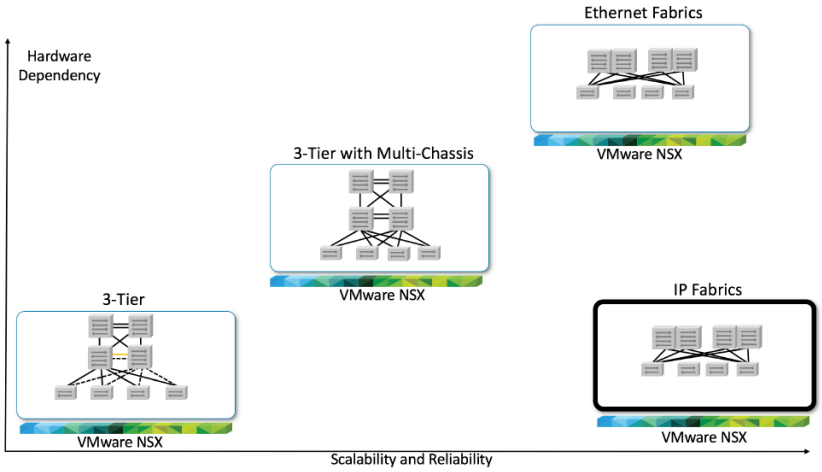


Figure 6-9 Data Center Network Architecture Evolution and VMware NSX

As they rely on STP - standardized on IEEE 802.1d, IEEE 802.1s, and IEEE 802.1w - three-tier designs do not impose hardware restriction for the switches that comprise them.

Three-tier topologies with multi-chassis link aggregation require that the same vendor, platform, firmware, and modules are implemented on each switch pair. Alternatively, an open design allows switch pairs from various vendors to participate in the topology.

And covered previously, L2 fabrics allow scalability and reliability, but are dependent on specific hardware.

VMware NSX can help deploy agility, security, extension, and automation to any of these structures, as Figure 6-9 portrays. This network virtualization platform also enables the use of an open physical L3 fabric to provision any type of connectivity required by modern applications with both scalability and reliability.

Astute data center network architects usually employ both physical and virtual elements as integrated parts of a consolidated data center network. These are the network professionals already working on the challenges that will define the next step of evolution of data center networks.

Consolidated Network Operations

It can be troubling to see how often network operations is treated as an afterthought. Many network professionals expect that networks should simply continue to work properly after the right configuration is issued, and if a problem does arise, no graphical interface could ever replace command outputs, debug messages, and keyboard spanking.

Among the many changes in networking since the beginning of this decade, data center network operations have received a greater focus as agility is not only required for application provisioning, but also for faster problem resolution, proper security forensics, anomaly detection, and appropriate capacity planning.

Effective network management and monitoring requires operational processes working in harmony with tools that can analyze physical and virtual elements in a consolidated approach. NSX offers native tools and protocols that can easily be integrated into the typical network operations toolbox, such as:

- **Command-Line Interface (CLI):** Centralized on NSX Manager and also present on many NSX constructs such as distributed logical routers and NSX Edges
- **Syslog:** Standardized messages that can be sent to remote servers compatible with this protocol
- **IP Flow Information Export (IPFIX) and NetFlow:** Provides information about flows traversing NSX elements. This information based on tuples containing source and destination IP addresses, transport protocol source and destination ports, among other data. These can be sent to IPFIX or NetFlow collectors.
- **Traffic mirroring:** Enables the copy of actual Ethernet frames or IP packets to a VM directly connected to NSX or to a traffic analyzer reachable via IP
- **Simple Network Management Protocol (SNMP):** Allows traps to be sent to SNMP receivers for proper analysis

Note: Please refer to the latest VMware documentation to verify which of these management features are available in any given NSX version. In the case of NSX for vSphere, some of these features are also implemented by the vSphere Distributed Switch, which is deeply integrated into the VMware NSX architecture as explained in Chapter 2.

As the next sections will present, NSX also provides native integration with two monitoring platforms that allow deep analysis and analytics of both physical and virtual network devices in a manner that was previously unavailable in the history of modern networking.

VMware vRealize Log Insight

It is standard practice for a solution to send logs based on events and errors for a later analysis. Handling these logs in a meaningful way is particularly challenging during periods of troubleshooting, especially with the ever-increasing amount of managed devices installed in modern data center environments.

While more traditional log management tools focus on specific devices and are not capable of generating insights from the analysis of received logs, VMware vRealize Log Insight (vRLI) enables operations professionals to make sense of log data received from physical, virtual, and cloud elements.

Log Insight receives, stores, and analyzes a myriad types of log data (e.g., application logs, configuration files, messages, dumps, among others) and provides predictive analytics, machine-learning algorithms, and root cause analysis tools for efficient troubleshooting.

A scalable log analysis tool, vRLI has an intuitive GUI that automatically adjusts the best viewing experience for a set of data. The platform sends alerts related to potential issues based on future analysis while also delivering real-time monitoring.

VMware vRealize Log Insight is extensible to third-party solutions via *Content Packs* that include:

- Applications such as Apache Tomcat, MongoDB, Oracle Database, Microsoft Active Directory, Microsoft Exchange, Microsoft IIS, Microsoft SharePoint, Microsoft SQL Server, and NGNIX
- Physical switches from many vendors such as Arista, Brocade, Cisco, Dell EMC, Juniper, and Lenovo
- Servers such as Cisco, Dell, and HP
- Operating systems such as Linux and Microsoft Windows
- Storage solutions from NetApp, Nimble, Dell EMC, Hitachi, and Pure Storage
- Firewalls from various vendors such as Cisco and Palo Alto Networks
- Physical server load balancers including F5 and Citrix NetScaler
- Cloud platforms as Pivotal Cloud Foundry and OpenStack
- VMware solutions such as Site Recovery Manager, vCloud Director, VMware vRealize® Operations Manager™, VMware Horizon®, VSAN, vSphere, vRealize Automation, and vRealize Orchestrator

Note: More information on vRealize Log Insight management packs is available at https://marketplace.vmware.com/vsx/

vRealize Log Insight also has management packs for NSX. Through this integration, vRLI provides reporting and alert visibility for every NSX construct and elements via pre-configured dashboards.

Figure 6-10 depicts some of the out-of-the-box dashboards vRLI can offer to a VMware NSX administrator.

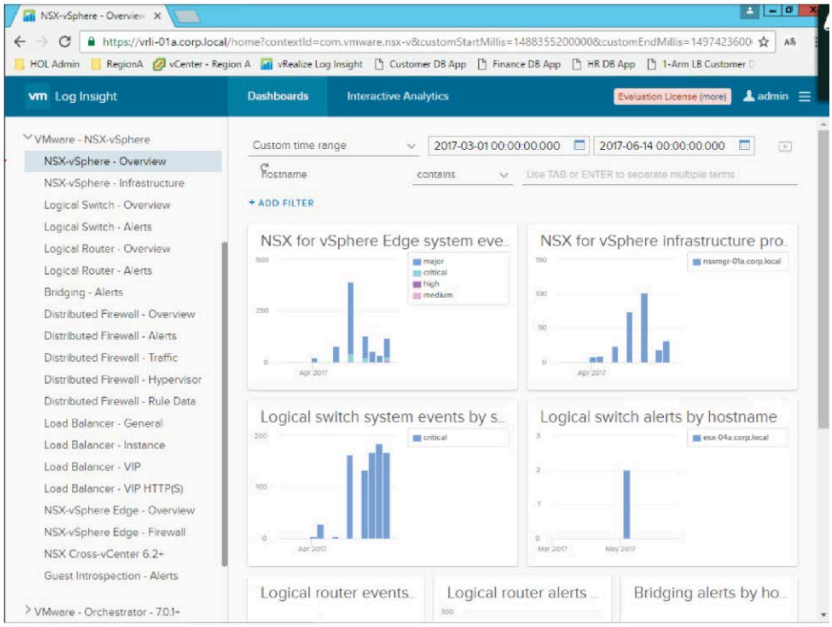


Figure 6-10 vRealize Log Insight VMware NSX Dashboards

Figure 6-10 shows the overview dashboard for NSX for vSphere, where it visualizes data such as NSX Edge system events, NSX infrastructure problems, logical switch system events and alerts, logical routers events and alerts, distributed firewall events and alerts, and many others.

All NSX dashboards can be configured to analyze a specific time range in order to help troubleshooting and root cause analysis operations. More detailed dashboards are also available, as Figure 6-11 exemplifies.

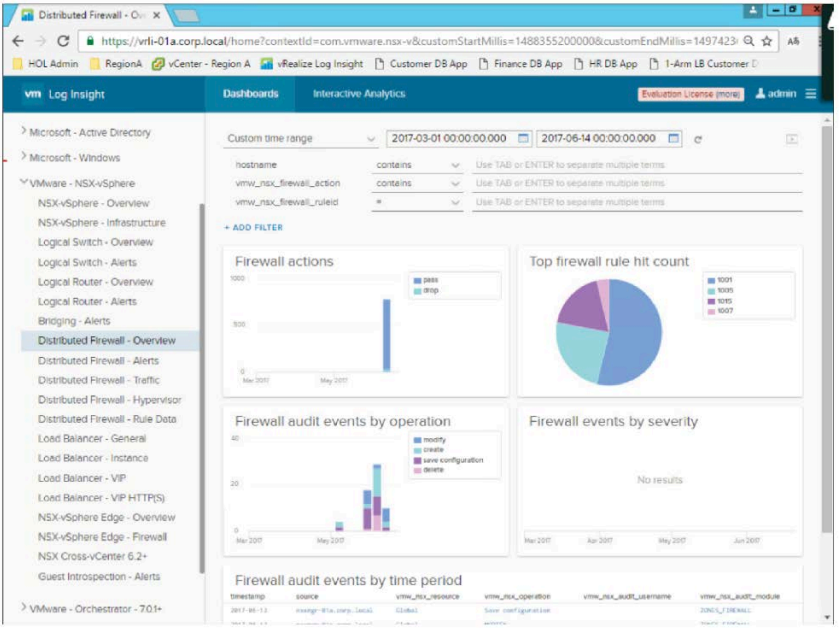


Figure 6-11 vRealize Log Insight VMware NSX Distributed Firewall Dashboards

Figure 6-11 depicts readily available dashboards related to distributed firewall such as firewall actions, top firewall rule hit count, firewall audit events by operation, firewall events by severity, and firewall audit events by time period among others.

As shown at the top of the figure, vRLI also allows the use of filters to narrow down results displayed by hostname, firewall rule action, and firewall rule identifier. Additional filters can be added to gather even more specific detail. On the bottom of Figure 6-11, audit events can be filtered by operation such as creation, deletion, or update of firewall rules.

In addition to the dashboards shown in Figures 6-10 and 6-11, vRealize Log Insight provides the following default dashboards for NSX, as shown on the left side of both figures:

- **Load balancer - general, instance, VIP, VIP HTTP(S)**
- **NSX Cross-vCenter**
- **Guest Introspection alerts**

vRealize Log Insight also allows interactive analytics - which show logs in real time, isolating all logs related to an event – along with the construction of customized dashboards.

Note: vRealize Log Insight for NSX is included with every acquired license of NSX.

VMware vRealize Network Insight

For data center networks that invested in network monitoring solutions, device event monitoring and bandwidth measurement were sometimes the only easily accessible outcome. With these scarce metrics, network designers could not do much more than perform basic extrapolations to improve the next hardware upgrade process.

Organizations that required finer traffic analysis (e.g., compliance mandates) were many times forced to build a parallel monitoring network to handle mirrored traffic. This was generated from traffic mirroring deployed on switches or, due to the limitations of these devices, network taps.

The devices comprising this network were not standard switches, but packet monitoring matrix switches that connect to all taps, traffic mirroring destination ports, and monitoring systems. Because these devices can direct, duplicate, and filter mirrored traffic to these systems, they are considered an expensive solution that is restricted to a few financial organizations.

IPFIX and NetFlow collectors are also popular among many organizations. While these systems can gather flows that traverse network devices via standard protocols, these collectors are not as used as traffic analysis tools because:

- Not all network devices support IPFIX or Netflow, which makes it hard to analyze all traffic that traverses a network if any physical or virtual device is missing.
- These software collectors can only analyze IP addresses and transport protocol ports, because they usually lack context for applications and other logical definitions such as server virtualization objects.
- Not many collectors understand network topology well enough to consolidate the information about the same flow detected on different network devices.

For many years, data center networks were similar to sports cars without a dashboard, while they continued to get faster with time.

As discussed in the previous chapters, network virtualization facilitated provisioning, security, extension, and automation. However, its centralized management and distribution of network functions untapped a potential for traffic data analysis that were never before possible.

vRealize Network Insight (vRNI) is the response to the challenge of traffic visibility of data center networks. Capable of receiving, storing, analyzing and providing insights about physical, virtual, and cloud traffic, vRNI is specially recommended for three main use cases: security planning, troubleshooting, and NSX operations.

In order to provide *security planning* and effectively help organizations deploy micro-segmentation according to the principles of Zero Trust architecture, vRNI offers full flow visibility in a data center network. The solution gathers IPFIX or NetFlow flow information from vDS, NSX, or physical network devices to provide detailed dashboards such as the one depicted in Figure 6-12.

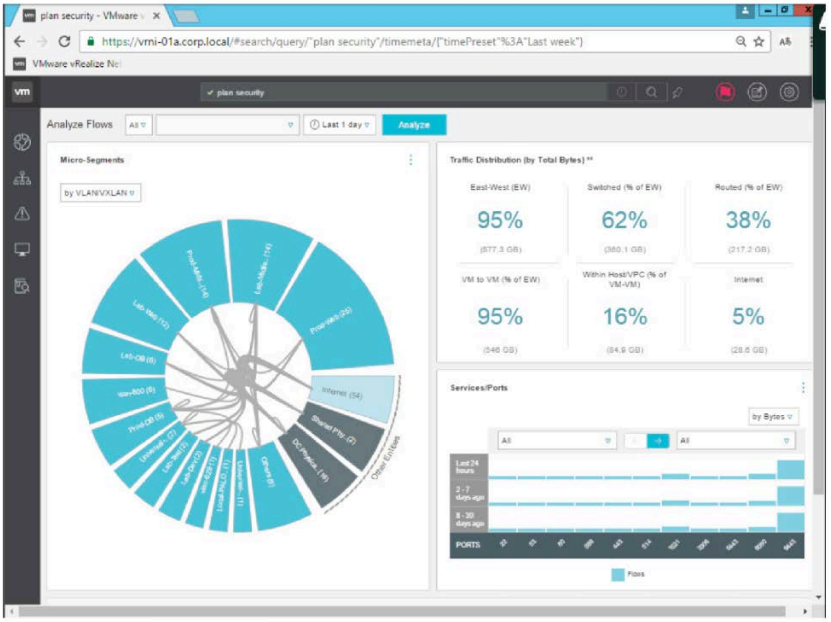


Figure 6-12 vRealize Network Insight Plan Security Dashboard

This example portrays a summary of all traffic flows observed in the last 24 hours. On the left side of the figure, the circular drawing called "Micro-Segments" represents the traffic distribution through lines connecting different VLANs or logical switches, here called VXLANs. This graphical representation can also be divided into other source and destination classifications such as application, application tier, subnet, folder, cluster, VM, port, security tag, security group, IP set, and Amazon Web Services Virtual Private Cloud (VPC).

This flexibility points to a key differentiator of vRNI over the traditional flow collectors available in the market: the deep network context it collects from different solutions such as vCenter, NSX, and physical network and security devices.

On the right upper side of Figure 6-12, "Traffic Distribution" introduces some insights about flows gathered during the same period, which are:

- **East-west traffic:** Represents traffic between hosts connected to the internal analyzed network
- **Switched (% of EW):** Translates the proportion of traffic volume between hosts in the same subnet
- **Routed (% of EW):** Corresponds to the traffic between hosts located in different subnets
- **VM to VM (% of EW):** Characterizes the percentage of traffic between virtual machines
- **Within Host/VPC (% of VM-VM):** Describes the percentage of VM-to-VM traffic that occurs within each host, which would not be detectable by physical network devices
- **Internet:** North-south traffic defined via public addresses but that can be customized if needed

The lower right table "Service/Ports" depicts a distribution of TCP and UDP ports used in different time ranges - last 24 hours, 2-7 days ago, 8-30 days ago – providing a quick reference of traffic behavior that can characterize an unpredicted peak or an attack.

All the graphs and insights are clickable, allowing a finer analysis of subsets of flows, such as the flows from a single VLAN/VXLAN segment, overall east-west traffic, and transport protocol destination port - respectively from the "Micro-Segments", "Traffic Distribution", and "Service/Ports".

Figure 6-13 charts the results from vRNI analyzing all flows detected on logical switch **Prod-Web**.

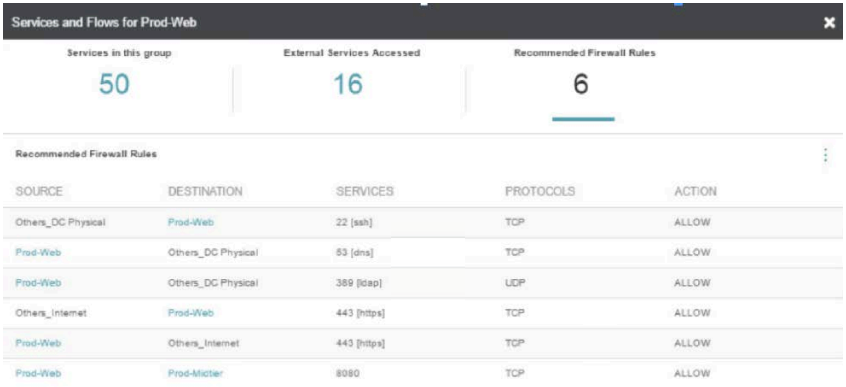


Figure 6-13 Services and Flows for Prod-Web

This image shows that the logical switch has transported 50 different services as defined by distinct transport protocol destination port and accessed 16 external services defined by IP addresses and destination ports. It also offers a recommendation of 6 distributed firewall rules that should be applied to implement Zero Trust on this segment.

Note: vRNI also helps improve security planning of NSX deployments through easy visualization of security groups, firewall rule tracking including compatible third-party firewalls, and user-defined events and notifications.

Figure 6-14 shows an example of how vRNI can improve the network and security troubleshooting process.

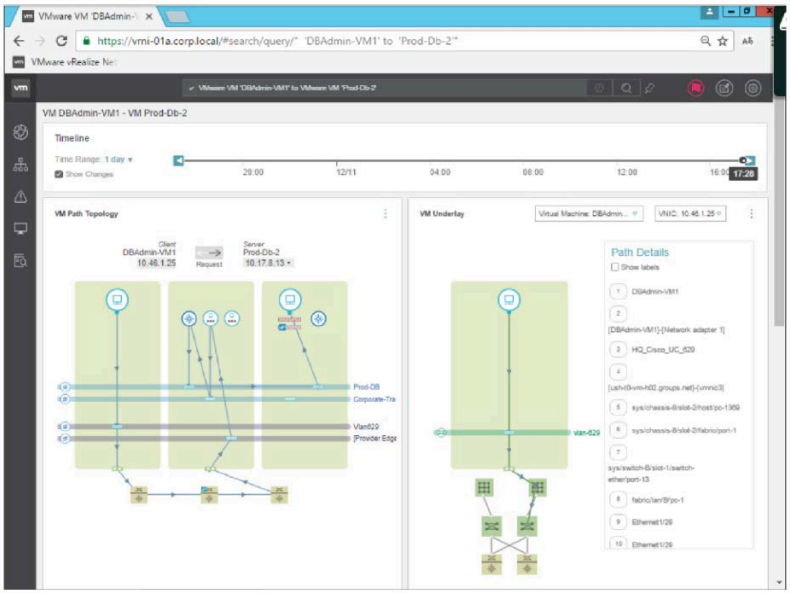


Figure 6-14 vRealize Network Insight Path Visibility

On the left side of the figure, the "VM Path Topology" uncovers all hops that comprise the communication between two VMs - **DBAdmin-VM1** and **Prod-Db-2** - including: VLANs, logical switches, physical devices, NSX Edges, distributed logical routers, distributed firewall, and third-party solutions integrated via network introspection.

Besides showing a simple and elegant design (e.g., each rectangle represents a host), the "VM Path Topology" graph allows the analysis of traffic on both directions, as indicated by both arrows at the top of the Request label. It allows the access to additional information and details of each element in the topology including VMs, physical hosts, VLAN, logical switches, switch ports, and VRFs among many others.

The right side of the image displays the "VM Underlay" graph, which depicts the same path topology, but only uncovers physical elements such as VLANs, switches, routers, and firewalls. Both graphs can show alerts when there are failures detected on an element that makes part of the path, be it logical or physical.

vRNI can also manage and scale network analytics for multiple NSX Managers through illustrative topologies and health graphs.

Figure 6-15 depicts how vRNI can provide *advanced management operations for NSX deployments*.

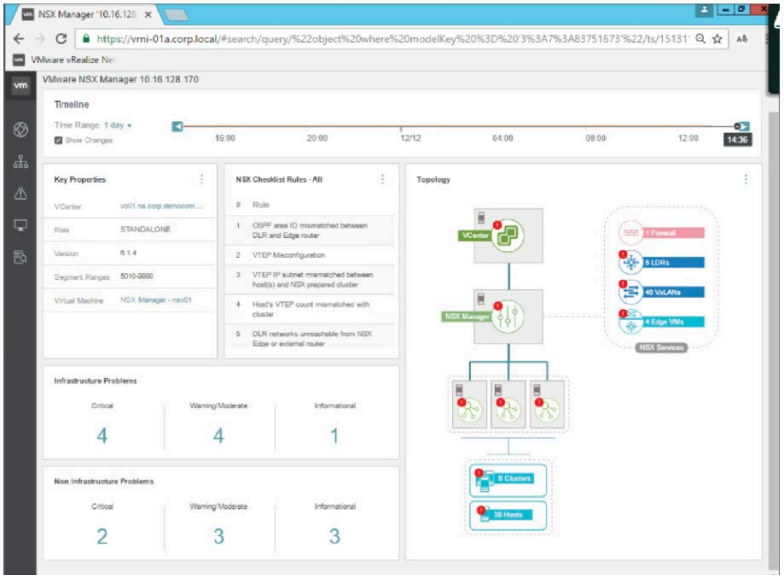


Figure 6-15 vRealize Network Insight NSX Advanced Management Operations

As shown in this figure, vRNI discloses a lot of information about NSX Manager 10.16.128.170, including:

- **Key Properties:** Associated vCenter; standalone, primary, secondary, or transit role; version; VXLAN segment ranges; and VM name
- **NSX Checklist Rules – All:** Describes detected misconfigurations based on best practices on NSX deployments
- **Topology:** Graphical representation of this NSX deployment with the numbers of problems related to the NSX infrastructure elements and NSX services
- **Infrastructure Problems:** Provides additional insight into discovered problems associated with VMware NSX infrastructural elements (e.g., vCenter, NSX Manager, NSX Controllers, clusters, hosts)
- **Non-Infrastructure Problems:** Offers classification of discovered problems associated with NSX services (e.g., firewall, DLRs, logical switches, and NSX Edges)

Besides allowing a finer analysis of any of the depicted elements or problems shown in this dashboard, vRNI also provides notification when changes happened via the timeline of problems, allowing selection of a specific time range.

All graphs and insights presented in this section only hint at the immense potential of vRNI for operators of data center networks. For a more complete experience with this solution, access the latest VMware vRNI hands-on lab available on http://labs.hol.vmware.com

What Did This Chapter Achieve?

This chapter introduced and explored the following concepts:

- **Evolution of Data Center Network Architectures:** In the mid-1990s, data center networks inherited architectural characteristics from campus networks, resulting on the popularization of three-tier designs with core, aggregation, and access switches. Since 2005, many data centers have leveraged multi-chassis link aggregation to improve bandwidth utilization and minimize the effect of STP in these environments. In the early 2010s, network vendors started to offer Ethernet fabrics in spine-leaf topologies as the next evolution step for data center network architectures in terms of reliability and scalability of L2 domains.

- **Data Center Consolidation with VMware NSX:** Regardless of their origins - multiple pairs of aggregation switches in a three-tier design or security segment isolation - NSX can securely abstract L2 silos in physical networks through logical switching, distributed logical routing, and distributed firewall. A network virtualization and security platform like NSX can free virtual machines from their original network silos to use computing capacity available in any other L2 domain in order to unlock and optimize resource utilization within or across data centers.

- **An Open Data Center Network Architecture:** Many customers still struggle with the decision of migrating to an Ethernet fabric topology, which significantly restricts future network devices to the same vendor and platform. NSX physical network requirements and its hardware VTEP compatibility list present an open alternative. By deploying this network virtualization platform, data center network architects can leverage simple L3 spine-leaf fabrics to transport all types of traffic without manufacturer lock-in.

- **Consolidated Network Operations:** NSX provides management information for existing monitoring tools through features such as syslog, IPFIX, NetFlow, port mirroring, and SNMP. Specialized VMware monitoring platforms such as vRealize Log Insight and vRealize Network Insight offer tight integration with the network virtualization platform while also monitoring the physical network and security devices. With such breadth, the VMware network and security solutions portfolio enables the unique experience of a consolidated data center network that seamlessly incorporate both virtual and physical elements.

In Conclusion

Network virtualization is a firm reality for more than three thousand organizations and eight thousand certified professionals around the world. And as more corporations and individuals join these ranks, it is impossible to deny that the flight of network and security with the wings of software has only begun.

Even for an old data center engineer, it is very exciting to behold the recent developments of the network virtualization platform that were unimaginable a few years ago.

Figure 6-16 illustrates how *Virtual Cloud Network (VCN)*- VMware's vision for current and future networking and security challenges – builds on the network virtualization foundations already established by NSX.

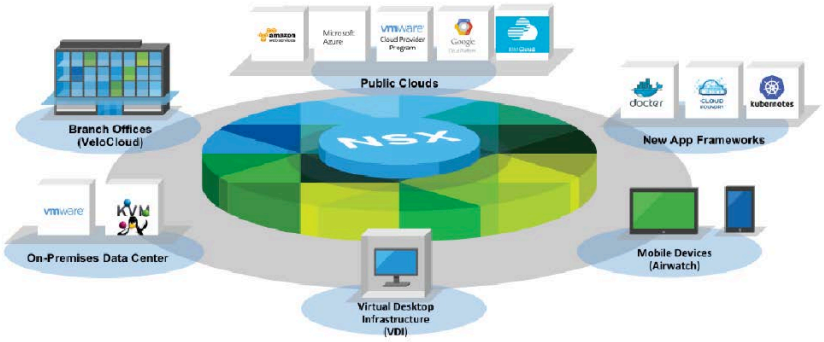


Figure 6-16 VMware Virtual Cloud Network

Throughout the many points in information technology depicted in Figure 6-16, Virtual Cloud Network has achieved the following breakthroughs:

- **On-Premises Data Center:** VMware vSphere-optimized and multi-hypervisor environments can also enjoy the benefits of network virtualization with NSX-V and NSX-T, respectively.
- **Virtual Desktop Infrastructure (VDI):** These platforms can leverage logical load balancing, micro-segmentation, and guest introspection to improve component availability, internal security, and protection against malware.
- **Mobile Devices (VMware Airwatch®):** For corporations that use the VMware enterprise mobility management software, NSX can provide application load balancing, device-to-server traffic control, and VPNs per mobile app.
- **New App Frameworks:** NSX is increasing agility and security on container-based environments such as Docker, Pivotal Cloud Foundry, and Kubernetes.
- **Public Clouds:** Whereas NSX is widely deployed in many public clouds such as IBM Cloud and members of the VMware Cloud Provider Program, NSX Cloud has brought network virtualization and micro-segmentation to non-traditional hypervisor environments such as Amazon Web Services. VMware Cloud on AWS enables applications to run over a VMware Software Defined Data Center environment integrating vSphere, vSAN, and NSX - all optimized to run in a dedicated, elastic, bare-metal AWS infrastructure.
- **Branch Offices:** With the recent acquisition of SD-WAN vendor VeloCloud by VMware, NSX initiates its journey from the data center towards the edge of wide area networks.

Simplicity: the final frontier. These are the voyages of the network and security platform NSX. Its mission: To explore exciting new possibilities, to seek out new concepts and new use cases, to boldly go where no switch has gone before!

Index

During their digital transformation process, many IT organizations still struggle with traditional networking methods and security approaches. By successfully addressing these challenges in thousands of real-world implementations, VMware NSX has established itself as the leading network virtualization platform.

In this book, data center expert and author Gustavo A. A. Santana thoroughly examines the specific circumstances that created such challenges and explains how VMware NSX overcomes them. This context will help those who want to:

- Substantially improve data center security with the Zero Trust model while implementing micro-segmentation
- Smooth data center automation and make private cloud computing implementations possible
- Radically simplify data center physical network architectures and enable full traffic visibility over physical and virtual network devices

This book is an essential resource for any IT professional that wants to understand the full potential of an enterprise-class network virtualization platform such as VMware NSX.

About the Author

Gustavo A. A. Santana is the leader of the Software Defined Data Center (SDDC) engineering team in VMware Latin America and the author of two books: Data Center Virtualization Fundamentals and CCNA Cloud CLDFND 210-451 Official Cert Guide.Throughout his 20 years of experience in the IT industry, Gustavo has worked on multiple data center projects that required extensive integration among different technology areas.A frequent speaker at VMware events and IT conferences, Gustavo holds the VMware Certified Implementation Expert in Network Virtualization (VCIX-NV) certification, is a triple Cisco Certified Internetwork Expert (CCIE), and an SNIA Certified Storage Networking Expert (SCSN-E).

vmware PRESS

Cover design: VMware

Cover photo: iStock / chombosan

www.vmware.com/go/run-nsx

ISBN-10: 0-9986104-7-X
ISBN-13: 978-0-9986104-7-4

ISBN 9780998610474
51299 >
9 780998 610474

$12.99